# Contents

#### [Figures](#_bookmark0) [xi](#_bookmark0)

#### [Tables](#_bookmark1) [xiii](#_bookmark1)

#### [About this book](#_bookmark2) [xv](#_bookmark2)

[Who should read this book](#_bookmark2) [xv](#_bookmark2)

[Typographical conventions](#_bookmark2) [xv](#_bookmark2)

[Related information](#_bookmark3) [xvii](#_bookmark3)

[Contacting IBM Customer Support for Rational](#_bookmark4) [software products](#_bookmark4) [xviii](#_bookmark4)

#### [Summary of changes](#_bookmark5) [xix](#_bookmark5)

[Automounter maps and mount points](#_bookmark20) [15](#_bookmark20)

[Specifying nonstandard mount points](#_bookmark20) [15](#_bookmark20)

[Specifying a nonstandard mount directory . .](#_bookmark21) . [16](#_bookmark21)

#### [Chapter 2. Licensing and license](#_bookmark22) [administration](#_bookmark22) [17](#_bookmark22)

[ClearCase Licensing and Rational Common](#_bookmark22)

. [17](#_bookmark22)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Licensing . . . . . .](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) |
| [ClearCase Licensing . .](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) | [.](#_bookmark22) |
| [License priorities . .](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) |
| [License report utility .](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) |
| [Specifying a license server host .](#_bookmark23) | | | | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) |
| [Setting up a license server . . .](#_bookmark23) | | | | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) |
| [The license database . . . .](#_bookmark23) | | | | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) | [.](#_bookmark23) |

. [17](#_bookmark22)

. [18](#_bookmark23)

. [18](#_bookmark23)

. [18](#_bookmark23)

. [18](#_bookmark23)

. [18](#_bookmark23)

#### [Chapter 1. Rational ClearCase network](#_bookmark6)

#### [planning and administration](#_bookmark6) [1](#_bookmark6)

[Overview of administrative responsibilities](#_bookmark6) [1](#_bookmark6)

[The Rational ClearCase product family](#_bookmark6) [1](#_bookmark6)

[Differences between Rational ClearCase and](#_bookmark7) [Rational ClearCase LT](#_bookmark7) [2](#_bookmark7)

[Hosts and host administration](#_bookmark7) [2](#_bookmark7)

[Release hosts](#_bookmark7) [2](#_bookmark7)

[Rational ClearCase hosts](#_bookmark7) [2](#_bookmark7)

[Rational ClearCase product family hosts](#_bookmark8) [3](#_bookmark8)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [Licensing and license administration](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [. 4](#_bookmark9) |
| [Data storage and data administration](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [. 4](#_bookmark9) |
| [The Rational ClearCase registry .](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [.](#_bookmark9) | [. 4](#_bookmark9) |

[Versioned Object Bases (VOBs)](#_bookmark9) [4](#_bookmark9)

[Views](#_bookmark10) [5](#_bookmark10)

[User and group administration](#_bookmark11) [6](#_bookmark11)

[Rational ClearCase server processes](#_bookmark11) [6](#_bookmark11)

[albd\_server](#_bookmark12) [7](#_bookmark12)

[Port assignment](#_bookmark13) [8](#_bookmark13)

[admin\_server](#_bookmark13) [8](#_bookmark13)

[credmap\_server](#_bookmark13) [8](#_bookmark13)

[view\_server](#_bookmark13) [8](#_bookmark13)

[vob\_server](#_bookmark14) [9](#_bookmark14)

[db\_server](#_bookmark14) [9](#_bookmark14)

[vobrpc\_server](#_bookmark14) [9](#_bookmark14)

[lockmgr](#_bookmark14) [9](#_bookmark14)

[rwp](#_bookmark15) [10](#_bookmark15)

[Server logs](#_bookmark15) [10](#_bookmark15)

[Logging on Linux and the UNIX system](#_bookmark15) [10](#_bookmark15)

[Logging on Windows](#_bookmark16) [11](#_bookmark16)

[Starting and stopping Rational ClearCase](#_bookmark16) [11](#_bookmark16)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [On Linux and the UNIX system](#_bookmark16) | [.](#_bookmark16) | [.](#_bookmark16) | [.](#_bookmark16) | [.](#_bookmark16) | [.](#_bookmark16) |
| [On Windows . . . . . . .](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) |
| [Starting and stopping the MVFS](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) | [.](#_bookmark17) |

. [11](#_bookmark16)

. [12](#_bookmark17)

. [12](#_bookmark17)

[Rational ClearCase and network-attached storage](#_bookmark17) [devices](#_bookmark17) [12](#_bookmark17)

[Configuring network access to the NAS device](#_bookmark18) [13](#_bookmark18)

[If you use DHCP](#_bookmark19) [14](#_bookmark19)

[If a Rational ClearCase host has multiple network](#_bookmark19) [interfaces](#_bookmark19) [14](#_bookmark19)

[Rational ClearCase and the NFS automounter . .](#_bookmark20) . [15](#_bookmark20)

[Adding new licenses to an existing license server](#_bookmark24) [host](#_bookmark24) [19](#_bookmark24)

[Setting up additional license server hosts . .](#_bookmark24) . [19](#_bookmark24) [Moving licenses to another host](#_bookmark25) [20](#_bookmark25)

[Renaming a license server host](#_bookmark26) [21](#_bookmark26)

[License database format](#_bookmark26) [21](#_bookmark26)

[License set definition lines](#_bookmark26) [21](#_bookmark26)

[User priority lines](#_bookmark26) [21](#_bookmark26)

[Excluded user lines](#_bookmark26) [21](#_bookmark26)

[Audit-Enable line](#_bookmark27) [22](#_bookmark27)

[Time-Out line](#_bookmark27) [22](#_bookmark27)

[Rational Common Licensing for Rational ClearCase](#_bookmark27)

[and Rational ClearCase LT](#_bookmark27) [22](#_bookmark27)

[Floating and nodelocked licenses](#_bookmark27) [22](#_bookmark27)

[Temporary, permanent, and term licenses . .](#_bookmark28) . [23](#_bookmark28) [IBM Rational ClearCase, IBM Rational](#_bookmark28)

[ClearCase LT, and IBM Rational Suite licenses.](#_bookmark28) . [23](#_bookmark28) [Installing and configuring a license server](#_bookmark28) [23](#_bookmark28)

[Installing and configuring Rational ClearCase and](#_bookmark28) [Rational ClearCase LT Windows client licenses .](#_bookmark28) . [23](#_bookmark28)

[The Rational License Key Administrator](#_bookmark29) [24](#_bookmark29)

[Installing and configuring Rational ClearCase or](#_bookmark29) [Rational ClearCase LT licenses on clients running](#_bookmark29)

[Linux or the UNIX system](#_bookmark29) [24](#_bookmark29)

#### [Chapter 3. Administering the Rational](#_bookmark30) [ClearCase registry](#_bookmark30) [25](#_bookmark30)

[Differences between the Rational ClearCase and](#_bookmark30) [Rational ClearCase LT registries](#_bookmark30) [25](#_bookmark30)

. [25](#_bookmark30)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Registry data .](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) | [.](#_bookmark30) |
| [Objects . .](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) | [.](#_bookmark31) |
| [Tags . . .](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) |

. [26](#_bookmark31)

. [27](#_bookmark32)

. [27](#_bookmark32)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [VOB tags and VOB access](#_bookmark32) | | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) | [.](#_bookmark32) |
| [View tags and view access](#_bookmark33) | | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) |
| [Registry regions . . .](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) | [.](#_bookmark33) |
| [Other shared data . . .](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) |
| [Server storage locations](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) |
| [Site defaults . . . .](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) |
| [The registry server host . .](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) | [.](#_bookmark36) |

. [28](#_bookmark33)

. [28](#_bookmark33)

. [31](#_bookmark36)

. [31](#_bookmark36)

. [31](#_bookmark36)

. [31](#_bookmark36)

[Defining a Rational ClearCase registry host. .](#_bookmark37) . [32](#_bookmark37)

[To define a computer running Linux or the](#_bookmark37)

[UNIX system as a registry host](#_bookmark37) [32](#_bookmark37)

[To define a Windows computer as a registry](#_bookmark38)

. [33](#_bookmark38)

|  |  |  |
| --- | --- | --- |
| [host . . . . . . . . . . . .](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) |
| [Setting and using the registry password .](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) |
| [The registry client list . . . . . . .](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) |

. [33](#_bookmark38)

. [33](#_bookmark38)

[Guidelines for using multiple Rational ClearCase](#_bookmark38) [registries](#_bookmark38) [33](#_bookmark38)

[Assigning a Rational ClearCase host’s registry and](#_bookmark38)

[Choosing a text mode for a view](#_bookmark55) [52](#_bookmark55)

[Enabling interop text mode support in older](#_bookmark55)

[VOBs](#_bookmark55) [52](#_bookmark55)

[To determine whether a VOB supports interop](#_bookmark56) [text modes](#_bookmark56) [53](#_bookmark56)

[Special procedure for Rational ClearCase](#_bookmark56)

[MultiSite users](#_bookmark56) [53](#_bookmark56)

[Differences in supported character sets](#_bookmark56) [53](#_bookmark56)

. [33](#_bookmark38)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [region . . . . . . . .](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) | [.](#_bookmark38) |
| [Managing registry data . .](#_bookmark39) | [.](#_bookmark39) | [.](#_bookmark39) | [.](#_bookmark39) | [.](#_bookmark39) | [.](#_bookmark39) | [.](#_bookmark39) | [.](#_bookmark39) |
| [Viewing objects and tags .](#_bookmark40) | [.](#_bookmark40) | [.](#_bookmark40) | [.](#_bookmark40) | [.](#_bookmark40) | [.](#_bookmark40) | [.](#_bookmark40) | [.](#_bookmark40) |
| [Registering a VOB or view](#_bookmark41) | [.](#_bookmark41) | [.](#_bookmark41) | [.](#_bookmark41) | [.](#_bookmark41) | [.](#_bookmark41) | [.](#_bookmark41) | [.](#_bookmark41) |
| [Creating, removing, or changing tags .](#_bookmark41) | | | | | [.](#_bookmark41) | [.](#_bookmark41) | [.](#_bookmark41) |
| [Changing site defaults . . . . . .](#_bookmark42) | | | | | [.](#_bookmark42) | [.](#_bookmark42) | [.](#_bookmark42) |

. [34](#_bookmark39)

. [35](#_bookmark40)

. [36](#_bookmark41)

. [36](#_bookmark41)

. [37](#_bookmark42)

[Adding a Rational ClearCase registry region .](#_bookmark42) . [37](#_bookmark42) [To create a new registry region](#_bookmark42) [37](#_bookmark42)

[To assign a host to a new registry region .](#_bookmark42) . [37](#_bookmark42) [To create tags in a new registry region . .](#_bookmark43) . [38](#_bookmark43)

[Creating VOBs and views in a Rational](#_bookmark43)

[ClearCase registry that has regions](#_bookmark43) [38](#_bookmark43)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [Removing a registry region . .](#_bookmark43) | [.](#_bookmark43) | [.](#_bookmark43) | [.](#_bookmark43) | [.](#_bookmark43) | [.](#_bookmark43) |
| [Renaming a registry server host](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) |
| [Registry administration guidelines.](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) | [.](#_bookmark44) |

. [38](#_bookmark43)

. [39](#_bookmark44)

. [39](#_bookmark44)

[Rational ClearCase registry backup and switchover](#_bookmark45) [40](#_bookmark45)

[To define a backup registry host](#_bookmark45) [40](#_bookmark45)

[To define a computer running Linux or the](#_bookmark45)

[UNIX system as a backup registry host . .](#_bookmark45) . [40](#_bookmark45) [To define a Windows computer as a backup](#_bookmark46) [registry host](#_bookmark46) [41](#_bookmark46)

[To promote a backup registry host to primary](#_bookmark46) [registry host](#_bookmark46) [41](#_bookmark46)

[To restore the primary registry host](#_bookmark47) [42](#_bookmark47)

[Moving the registry to a host not configured for](#_bookmark47) [registry backups](#_bookmark47) [42](#_bookmark47)

[**Chapter 4. Rational ClearCase network**](#_bookmark48)[**administration in mixed environments .**](#_bookmark48)[**45**](#_bookmark48)[Common user and group names](#_bookmark48) [45](#_bookmark48)

[How user and group names are compared by](#_bookmark48) [Rational ClearCase](#_bookmark48) [45](#_bookmark48)

[Credentials mapping](#_bookmark49) [46](#_bookmark49)

[Specifying the Windows domain for credential](#_bookmark49) [mapping of users on hosts running Linux or](#_bookmark49)

[the UNIX system](#_bookmark49) [46](#_bookmark49)

[Credentials for the Rational ClearCase server](#_bookmark49) [process user on Linux and the UNIX system](#_bookmark49) . [46](#_bookmark49)

. [47](#_bookmark50)

|  |  |  |  |
| --- | --- | --- | --- |
| [Cross-platform file system access . . .](#_bookmark50) | [.](#_bookmark50) | [.](#_bookmark50) | [.](#_bookmark50) |
| [The Rational ClearCase File Service .](#_bookmark50) | [.](#_bookmark50) | [.](#_bookmark50) | [.](#_bookmark50) |
| [Enabling CCFS on Windows hosts .](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) |

. [47](#_bookmark50)

. [48](#_bookmark51)

[CCFS and remote Windows view server hosts](#_bookmark51) [48](#_bookmark51)

#### [Chapter 5. VOB and view access](#_bookmark57)

#### [control](#_bookmark57) [55](#_bookmark57)

[Fundamentals of VOB and view access control .](#_bookmark57) . [55](#_bookmark57) [Users and groups](#_bookmark57) [55](#_bookmark57)

[The primary group](#_bookmark57) [55](#_bookmark57)

[Limitations when a user belongs to more than](#_bookmark58)

[32 groups](#_bookmark58) [56](#_bookmark58)

[Privileged users and groups](#_bookmark58) [56](#_bookmark58)

[The Rational ClearCase server process user .](#_bookmark58) . [56](#_bookmark58) [Restricted view access privileges for remote](#_bookmark58)

. [56](#_bookmark58)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [root . . . . . . .](#_bookmark58) | [.](#_bookmark58) | [.](#_bookmark58) | [.](#_bookmark58) | [.](#_bookmark58) | [.](#_bookmark58) | [.](#_bookmark58) | [.](#_bookmark58) |
| [User processes . . . .](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) |
| [Rational ClearCase objects](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) |
| [Protection modes . .](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) | [.](#_bookmark59) |

. [57](#_bookmark59)

. [57](#_bookmark59)

. [57](#_bookmark59)

[Access to VOB and view data](#_bookmark60) [58](#_bookmark60)

[Access control for VOBs and VOB objects](#_bookmark61) [59](#_bookmark61)

. [59](#_bookmark61)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [Access control for VOBs . . .](#_bookmark61) | [.](#_bookmark61) | [.](#_bookmark61) | [.](#_bookmark61) | [.](#_bookmark61) | [.](#_bookmark61) |
| [Permission to create VOBs .](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) |
| [Permission to delete VOBs .](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) |
| [Permission to read VOBs . .](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) |
| [Permission to write VOBs .](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) |
| [Permission to execute VOBs .](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) |
| [Access control for elements . .](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) | [.](#_bookmark62) |
| [Permission to create elements](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) |
| [Permission to delete elements](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) |
| [Permission to read elements .](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) |
| [Permission to write elements](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) | [.](#_bookmark63) |
| [Permission to execute elements .](#_bookmark64) | | [.](#_bookmark64) | [.](#_bookmark64) | [.](#_bookmark64) | [.](#_bookmark64) |

. [60](#_bookmark62)

. [60](#_bookmark62)

. [60](#_bookmark62)

. [60](#_bookmark62)

. [60](#_bookmark62)

. [60](#_bookmark62)

. [61](#_bookmark63)

. [61](#_bookmark63)

. [61](#_bookmark63)

. [61](#_bookmark63)

. [62](#_bookmark64)

[Access control for other VOB objects](#_bookmark64) [62](#_bookmark64)

[Permission to create other VOB objects . .](#_bookmark65) . [63](#_bookmark65) [Permission to delete other VOB objects . .](#_bookmark65) . [63](#_bookmark65) [Permission to read other VOB objects](#_bookmark65) [63](#_bookmark65)

[Permission to write other VOB objects . .](#_bookmark65) . [63](#_bookmark65) [Locks on VOB objects](#_bookmark65) [63](#_bookmark65)

[Locking type objects](#_bookmark65) [63](#_bookmark65)

[Access control for views and view objects](#_bookmark66) [64](#_bookmark66)

[Access control for dynamic views](#_bookmark66) [64](#_bookmark66)

. [65](#_bookmark67)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [Permission to create views](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) |
| [Permission to delete views](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) |
| [Permission to read views .](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) |
| [Permission to write views](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) | [.](#_bookmark67) |
| [Permission to execute views .](#_bookmark68) | | [.](#_bookmark68) | [.](#_bookmark68) | [.](#_bookmark68) | [.](#_bookmark68) | [.](#_bookmark68) |
| [ccess control for view-private files . . . .](#_bookmark68) | | | | | | |

. [65](#_bookmark67)

. [65](#_bookmark67)

. [65](#_bookmark67)

. [66](#_bookmark68)

[Network-attached storage and cross-platform file](#_bookmark51)

. [48](#_bookmark51)

[A](#_bookmark68)

[Initial owner, group, and protection mode on](#_bookmark68)

. [66](#_bookmark68)

. [49](#_bookmark52)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [access tools . . . . .](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) | [.](#_bookmark51) |
| [Case-sensitivity . . . . .](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) |
| [General recommendations](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) |
| [Case-sensitivity and the MVFS](#_bookmark52) | | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) | [.](#_bookmark52) |

. [49](#_bookmark52)

. [49](#_bookmark52)

[When to use a case-preserving MVFS](#_bookmark53) [50](#_bookmark53)

[Case-sensitivity in snapshot and Web views .](#_bookmark53) . [50](#_bookmark53)

. [50](#_bookmark53)

|  |  |  |
| --- | --- | --- |
| [Configuring text modes for views . . . .](#_bookmark53) | [.](#_bookmark53) | [.](#_bookmark53) |
| [Text modes . . . . . . . . . .](#_bookmark54) | [.](#_bookmark54) | [.](#_bookmark54) |
| [Determining the text mode for a view](#_bookmark55) | [.](#_bookmark55) | [.](#_bookmark55) |

. [51](#_bookmark54)

. [52](#_bookmark55)

[Linux and the UNIX system](#_bookmark68) [66](#_bookmark68)

[Initial owner, Group, and protection mode on](#_bookmark68)

. [66](#_bookmark68)

|  |  |  |
| --- | --- | --- |
| [Windows . . . . . . . . . .](#_bookmark68) | [.](#_bookmark68) | [.](#_bookmark68) |
| [Permission to create view-private files](#_bookmark69) | [.](#_bookmark69) | [.](#_bookmark69) |
| [Permission to delete view-private files](#_bookmark69) | [.](#_bookmark69) | [.](#_bookmark69) |
| [Permission to read view-private files .](#_bookmark69) | [.](#_bookmark69) | [.](#_bookmark69) |
| [Permission to write view-private files](#_bookmark69) | [.](#_bookmark69) | [.](#_bookmark69) |

. [67](#_bookmark69)

. [67](#_bookmark69)

. [67](#_bookmark69)

. [67](#_bookmark69)

[Permission to execute view-private files . .](#_bookmark70) . [68](#_bookmark70)

[Access control for derived objects](#_bookmark70) [68](#_bookmark70)

[Rational ClearCase and native file system](#_bookmark70)

[permissions](#_bookmark70) [68](#_bookmark70)

#### [Chapter 6. VOB administration](#_bookmark71) [71](#_bookmark71)

[Introduction to VOBs and VOB administration .](#_bookmark71) . [71](#_bookmark71) [Access to VOB data and metadata](#_bookmark71) [71](#_bookmark71)

[VOB attributes](#_bookmark72) [72](#_bookmark72)

[VOB schema versions](#_bookmark72) [72](#_bookmark72)

[Upgrading your VOB schema version . .](#_bookmark72) . [72](#_bookmark72) [VOB feature levels](#_bookmark72) [72](#_bookmark72)

[To display the VOB feature level](#_bookmark73) [73](#_bookmark73)

[To change the VOB feature level](#_bookmark73) [73](#_bookmark73)

[VOB server host configuration guidelines](#_bookmark74) [74](#_bookmark74)

[Physical memory](#_bookmark74) [74](#_bookmark74)

[Swap space](#_bookmark74) [74](#_bookmark74)

[Disk capacity](#_bookmark75) [75](#_bookmark75)

[Processor capacity](#_bookmark75) [75](#_bookmark75)

[Network connectivity](#_bookmark75) [75](#_bookmark75)

[Adjusting kernel resources on Linux and the](#_bookmark75)

[UNIX system](#_bookmark75) [75](#_bookmark75)

[Creating VOBs](#_bookmark75) [75](#_bookmark75)

[Access requirements for VOB storage](#_bookmark76) [76](#_bookmark76)

[Creating server storage locations for VOBs . .](#_bookmark76) . [76](#_bookmark76) [Creating server storage locations on a NAS](#_bookmark77)

. [77](#_bookmark77)

|  |  |  |  |
| --- | --- | --- | --- |
| [device . . . . . . . . . .](#_bookmark77) | [.](#_bookmark77) | [.](#_bookmark77) | [.](#_bookmark77) |
| [To create a VOB on a VOB server host](#_bookmark77) | [.](#_bookmark77) | [.](#_bookmark77) | [.](#_bookmark77) |
| [To create a VOB on a NAS device .](#_bookmark78) | [.](#_bookmark78) | [.](#_bookmark78) | [.](#_bookmark78) |
| [To create a VOB on a remote Windows host](#_bookmark79) | | | |
| [Public and private VOBs . . . . . .](#_bookmark79) | | [.](#_bookmark79) | [.](#_bookmark79) |
| [To create a public VOB . . . . .](#_bookmark80) | | [.](#_bookmark80) | [.](#_bookmark80) |
| [Linking a VOB to an administrative VOB](#_bookmark80) | | [.](#_bookmark80) | [.](#_bookmark80) |

. [77](#_bookmark77)

. [78](#_bookmark78)

[79](#_bookmark79)

. [79](#_bookmark79)

. [80](#_bookmark80)

. [80](#_bookmark80)

[Replacing a VOB server host for a NAS device .](#_bookmark80) . [80](#_bookmark80) [Troubleshooting VOB access problems](#_bookmark81) [81](#_bookmark81)

[Incorrect global path information](#_bookmark81) [81](#_bookmark81)

[The automounter on Linux or the UNIX system](#_bookmark81)

[does not use the hosts map](#_bookmark81) [81](#_bookmark81)

[Adjusting the VOB’s group ownership](#_bookmark82)

[information](#_bookmark82) [82](#_bookmark82)

[Enabling setuid execution on Linux and the](#_bookmark82)

[Creating remote storage pools on hosts running](#_bookmark90) [Linux or the UNIX system](#_bookmark90) [90](#_bookmark90)

[Changing elements’ source pool assignments .](#_bookmark91) . [91](#_bookmark91) [Example: assigning all files in a directory to a](#_bookmark91) [new pool](#_bookmark91) [91](#_bookmark91)

[Example: Moving an existing storage pool to](#_bookmark92) [another disk](#_bookmark92) [92](#_bookmark92)

#### [Chapter 7. VOB datatypes and](#_bookmark93) [administrative VOB hierarchies](#_bookmark93) [93](#_bookmark93)

[VOB datatypes](#_bookmark93) [93](#_bookmark93)

[The VOB object and replica objects](#_bookmark93) [93](#_bookmark93)

[File system objects](#_bookmark93) [93](#_bookmark93)

[Event records](#_bookmark94) [94](#_bookmark94)

[Shareable derived objects](#_bookmark94) [94](#_bookmark94)

[Configuration records](#_bookmark94) [94](#_bookmark94)

[Type objects](#_bookmark94) [94](#_bookmark94)

[Instances of type objects](#_bookmark95) [95](#_bookmark95)

[Scope of type objects](#_bookmark95) [95](#_bookmark95)

[Predefined and user-defined type objects .](#_bookmark96) . [96](#_bookmark96) [Changing an element’s type](#_bookmark96) [96](#_bookmark96)

[Administrative VOB hierarchies and global types.](#_bookmark96) . [96](#_bookmark96) [Administrative VOB hierarchies](#_bookmark96) [96](#_bookmark96)

[Listing an AdminVOB hyperlink](#_bookmark98) [98](#_bookmark98)

[Restrictions on administrative VOB hierarchies](#_bookmark99) [99](#_bookmark99)

[If an administrative VOB becomes unavailable](#_bookmark99) . [99](#_bookmark99) [Removing a VOB from an administrative VOB](#_bookmark100) [hierarchy](#_bookmark100) [100](#_bookmark100)

[Removing the AdminVOB hyperlink . . .](#_bookmark100) [100](#_bookmark100) [Removing all GlobalDefinition hyperlinks](#_bookmark100) [100](#_bookmark100)

[Removing an administrative VOB](#_bookmark101) [101](#_bookmark101)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [Working with global types . . .](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [101](#_bookmark101) |
| [Creating a global type . . .](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [.](#_bookmark101) | [101](#_bookmark101) |
| [Auto-Make-Type operations](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [102](#_bookmark102) |
| [Describing global types . . .](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [.](#_bookmark102) | [102](#_bookmark102) |
| [Listing global types . . . .](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [103](#_bookmark103) |
| [Listing the history of a global ty](#_bookmark103) | [pe](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [.](#_bookmark103) | [103](#_bookmark103) |

[Changing the protection of a global type . . .](#_bookmark104) [104](#_bookmark104)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [Locking or unlocking a global type](#_bookmark104) | | [.](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [104](#_bookmark104) |
| [Copying a global type . . .](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [.](#_bookmark104) | [104](#_bookmark104) |
| [Renaming a global type . . .](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [105](#_bookmark105) |
| [Changing the scope of a type .](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [.](#_bookmark105) | [105](#_bookmark105) |
| [Removing a global type . . .](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [106](#_bookmark106) |
| [Cleaning up global types . .](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [.](#_bookmark106) | [106](#_bookmark106) |

. [82](#_bookmark82)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [UNIX system . . . .](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) |
| [Making a VOB inaccessible](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) |
| [To lock a VOB . . .](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) | [.](#_bookmark82) |
| [To remove a VOB tag .](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) |
| [Removing a VOB . . .](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) |
| [The VOB storage directory](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) | [.](#_bookmark83) |
| [VOB storage pools . .](#_bookmark84) | [.](#_bookmark84) | [.](#_bookmark84) | [.](#_bookmark84) | [.](#_bookmark84) | [.](#_bookmark84) | [.](#_bookmark84) | [.](#_bookmark84) | [.](#_bookmark84) |
| [Source storage pools](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) |
| [Cleartext storage pools](#_bookmark85) | | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) |
| [Derived object storage pools.](#_bookmark85) | | | | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) | [.](#_bookmark85) |
| [VOB database . . . . .](#_bookmark86) | | | | [.](#_bookmark86) | [.](#_bookmark86) | [.](#_bookmark86) | [.](#_bookmark86) | [.](#_bookmark86) |
| [Preserved database subdirectories](#_bookmark87) | | | | | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) |
| [The .identity directory .](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) |
| [VOB storage management](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) |
| [Using the scheduler .](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) | [.](#_bookmark87) |
| [Monitoring VOB storage](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) |
| [Scrubbing . . . . .](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) | [.](#_bookmark88) |

. [82](#_bookmark82)

. [82](#_bookmark82)

. [83](#_bookmark83)

. [83](#_bookmark83)

. [83](#_bookmark83)

. [84](#_bookmark84)

. [85](#_bookmark85)

. [85](#_bookmark85)

. [85](#_bookmark85)

. [86](#_bookmark86)

. [87](#_bookmark87)

. [87](#_bookmark87)

. [87](#_bookmark87)

. [87](#_bookmark87)

. [88](#_bookmark88)

. [88](#_bookmark88)

[Removing unneeded versions from a VOB . .](#_bookmark88) . [88](#_bookmark88)

. [89](#_bookmark89)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [The lost+found directory . . . . .](#_bookmark89) | [.](#_bookmark89) | [.](#_bookmark89) | [.](#_bookmark89) | [.](#_bookmark89) |
| [Creating additional storage for a VOB](#_bookmark90) | [.](#_bookmark90) | [.](#_bookmark90) | [.](#_bookmark90) | [.](#_bookmark90) |
| [Creating additional storage pools .](#_bookmark90) | [.](#_bookmark90) | [.](#_bookmark90) | [.](#_bookmark90) | [.](#_bookmark90) |

. [90](#_bookmark90)

. [90](#_bookmark90)

[Replicated administrative VOB hierarchies . . .](#_bookmark106) [106](#_bookmark106) [Global types and mastership](#_bookmark108) [108](#_bookmark108)

[Changing mastership of a global type](#_bookmark109) [109](#_bookmark109)

#### [Chapter 8. Moving VOBs and](#_bookmark110)

#### [relocating VOB data](#_bookmark110) [111](#_bookmark110)

[Relocating elements to another VOB](#_bookmark110) [111](#_bookmark110)

[What does relocate do?](#_bookmark110) [111](#_bookmark110)

[Steps to take before relocating elements . .](#_bookmark111) . [112](#_bookmark111) [Handling borderline elements](#_bookmark111) [112](#_bookmark111)

[Handling errors during a relocate operation .](#_bookmark112) . [113](#_bookmark112) [Errors involving locked types or checked-out](#_bookmark112)

[files](#_bookmark112) [113](#_bookmark112)

[Errors while removing elements from the](#_bookmark112)

[source VOB](#_bookmark112) [113](#_bookmark112)

[After relocating elements](#_bookmark112) [113](#_bookmark112)

[Using update mode](#_bookmark113) [114](#_bookmark113)

[Moving VOBs](#_bookmark113) [114](#_bookmark113)

[Important steps to take when moving any VOB](#_bookmark114) [115](#_bookmark114)

[Special considerations for replicated VOBs](#_bookmark114) [115](#_bookmark114)

. [116](#_bookmark115)

|  |  |  |  |
| --- | --- | --- | --- |
| [Moving a VOB on Windows . . . .](#_bookmark115) | [.](#_bookmark115) | [.](#_bookmark115) | [.](#_bookmark115) |
| [Moving a VOB within a domain . .](#_bookmark115) | [.](#_bookmark115) | [.](#_bookmark115) | [.](#_bookmark115) |
| [Moving a VOB to a different domain](#_bookmark115) | [.](#_bookmark115) | [.](#_bookmark115) | [.](#_bookmark115) |

. [116](#_bookmark115)

. [116](#_bookmark115)

[Moving a VOB on Linux or the UNIX system .](#_bookmark117) . [119](#_bookmark117) [If the VOB has remote pools](#_bookmark117) [119](#_bookmark117)

[Consolidating remote pools](#_bookmark118) [120](#_bookmark118)

[If the VOB is exported for non-ClearCase access](#_bookmark119) [121](#_bookmark119)

[Moving a VOB between hosts running Linux or](#_bookmark119) [the UNIX system that have the same](#_bookmark119)

[architecture](#_bookmark119) [121](#_bookmark119)

[Moving a VOB between hosts running Linux or](#_bookmark120) [the UNIX system that have different](#_bookmark120)

[architectures](#_bookmark120) [122](#_bookmark120)

[Moving a VOB to a different operating system type](#_bookmark120) [122](#_bookmark120) [Moving a VOB from Windows to Linux or the](#_bookmark121)

[UNIX system](#_bookmark121) [123](#_bookmark121)

[Moving a VOB from Linux or the UNIX system](#_bookmark123)

[to Windows](#_bookmark123) [125](#_bookmark123)

[Moving a VOB to network-attached storage . . .](#_bookmark125) [128](#_bookmark125) [Moving a VOB that has no remote pools . . .](#_bookmark125) [128](#_bookmark125) [Moving a VOB that has remote pools](#_bookmark125) [128](#_bookmark125)

#### [Chapter 9. View administration](#_bookmark126) [129](#_bookmark126)

[Introduction to views and view administration . .](#_bookmark126) [129](#_bookmark126) [Dynamic views](#_bookmark126) [129](#_bookmark126)

[The view root](#_bookmark127) [130](#_bookmark127)

[To change the default view root](#_bookmark127) [130](#_bookmark127)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [The view storage directory .](#_bookmark127) | | [.](#_bookmark127) | [.](#_bookmark127) | [.](#_bookmark127) | [.](#_bookmark127) | [.](#_bookmark127) | [.](#_bookmark127) | [.](#_bookmark127) | [130](#_bookmark127) |
| [View-Private storage . . .](#_bookmark128) | | [.](#_bookmark128) | [.](#_bookmark128) | [.](#_bookmark128) | [.](#_bookmark128) | [.](#_bookmark128) | [.](#_bookmark128) | [.](#_bookmark128) | [131](#_bookmark128) |
| [View database . . . . .](#_bookmark129) | | [.](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [132](#_bookmark129) |
| [How a dynamic view selects](#_bookmark129) | | [ver](#_bookmark129) | [sio](#_bookmark129) | [ns](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [.](#_bookmark129) | [132](#_bookmark129) |
| [How a dynamic view manages derived objects](#_bookmark129) [132](#_bookmark129) | | | | | | | | | |
| [The Multiversion File System . .](#_bookmark130) | | | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [133](#_bookmark130) |
| [Supported file types . . . .](#_bookmark130) | | | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [.](#_bookmark130) | [133](#_bookmark130) |
| [The MVFS and audited builds.](#_bookmark131) | | | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [134](#_bookmark131) |
| [Known limitations of the MVFS](#_bookmark131) | | | [on](#_bookmark131) | [W](#_bookmark131) | [ind](#_bookmark131) | [ow](#_bookmark131) | [s](#_bookmark131) |  | [134](#_bookmark131) |
| [The MVFS and case-sensitivity .](#_bookmark131) | | | | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [134](#_bookmark131) |
| [Running executables in the MVFS](#_bookmark131) | | | | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [.](#_bookmark131) | [134](#_bookmark131) |
| [MVFS performance . .](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [135](#_bookmark132) |
| [Snapshot views. . . . .](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [135](#_bookmark132) |
| [Snapshot view directory .](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [.](#_bookmark132) | [135](#_bookmark132) |
| [View storage directory .](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [136](#_bookmark133) |
| [View database . . . .](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [.](#_bookmark133) | [136](#_bookmark133) |

[How client programs locate snapshot views . .](#_bookmark134) [137](#_bookmark134) [How a snapshot view selects versions](#_bookmark134) [137](#_bookmark134)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [Creating a view . . . . . . .](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [137](#_bookmark134) |
| [Remote view storage . . . . .](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [.](#_bookmark134) | [137](#_bookmark134) |
| [Creating a view on a NAS device](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [138](#_bookmark135) |
| [Moving a view . . . . . . . .](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [.](#_bookmark135) | [138](#_bookmark135) |

[Determining whether a dynamic view on Linux](#_bookmark135)

[or the UNIX system has remote private storage .](#_bookmark135) [138](#_bookmark135) [Moving a view to a host with the same](#_bookmark135)

[architecture or to a NAS device](#_bookmark135) [138](#_bookmark135)

[Special procedures for moving snapshot](#_bookmark136)

[views on Windows](#_bookmark136) [140](#_bookmark136)

[Moving a view to a host running Linux or the](#_bookmark137)

[UNIX system with a different architecture . . .](#_bookmark137) [141](#_bookmark137)

[Replacing a view server host for a NAS device . .](#_bookmark138) [142](#_bookmark138)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [Displaying the properties of a view .](#_bookmark139) | | | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) [143](#_bookmark139) |
| [Taking a view out of service .](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) [143](#_bookmark139) |
| [Restoring the view to service](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) [143](#_bookmark139) |
| [Removing a view . . . . .](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) | [.](#_bookmark139) [143](#_bookmark139) |

#### [Chapter 10. Backing up critical](#_bookmark141)

#### [Rational ClearCase data](#_bookmark141) [145](#_bookmark141)

[Requirements for VOB and view backup tools . .](#_bookmark141) [145](#_bookmark141) [VOB backup strategies](#_bookmark141) [145](#_bookmark141)

[Choosing between standard and snapshot](#_bookmark142)

[backups](#_bookmark142) [146](#_bookmark142)

[Backing up VOBs with vob\_snapshot](#_bookmark143) [147](#_bookmark143)

[Deferred deletion of source containers](#_bookmark143) [147](#_bookmark143)

[Backing up VOBs with standard backup](#_bookmark144)

[procedures](#_bookmark144) [148](#_bookmark144)

[Determining the location of the VOB storage](#_bookmark144)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [directory . . . . . . . . . .](#_bookmark144) | | | | | [.](#_bookmark144) | [.](#_bookmark144) | [.](#_bookmark144) | [.](#_bookmark144) | [148](#_bookmark144) |
| [Locking and unlocking the VOB . .](#_bookmark144) | | | | | [.](#_bookmark144) | [.](#_bookmark144) | [.](#_bookmark144) | [.](#_bookmark144) | [148](#_bookmark144) |
| [If the VOB has remote storage pools.](#_bookmark145) | | | | | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [149](#_bookmark145) |
| [What to back up if you cannot back up](#_bookmark145) | | | | | | | | | |
| [everything . . . . .](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [149](#_bookmark145) |
| [DO pool backup . .](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [.](#_bookmark145) | [149](#_bookmark145) |
| [Cleartext pool backup](#_bookmark146) | [.](#_bookmark146) | [.](#_bookmark146) | [.](#_bookmark146) | [.](#_bookmark146) | [.](#_bookmark146) | [.](#_bookmark146) | [.](#_bookmark146) | [.](#_bookmark146) | [150](#_bookmark146) |

[Administrative directory backup](#_bookmark146) [150](#_bookmark146)

[Do not perform incremental backups](#_bookmark146) [150](#_bookmark146)

[Backing up a view](#_bookmark147) [151](#_bookmark147)

[Backing up Rational ClearCase registry data . . .](#_bookmark148) [152](#_bookmark148) [Backing up related databases together](#_bookmark148) [152](#_bookmark148)

[Determining database relationships](#_bookmark150) [154](#_bookmark150)

#### [Chapter 11. Restoring critical Rational](#_bookmark151) [ClearCase data](#_bookmark151) [155](#_bookmark151)

[Before you restore](#_bookmark151) [155](#_bookmark151)

[Restoring a VOB from backup with vob\_restore](#_bookmark151) [155](#_bookmark151)

[Scenarios supported by vob\_restore](#_bookmark152) [156](#_bookmark152)

[Running vob\_restore](#_bookmark153) [157](#_bookmark153)

[Restoring a VOB from backup without vob\_restore](#_bookmark155) [159](#_bookmark155)

[Restoring a VOB on network-attached storage . .](#_bookmark157) [161](#_bookmark157) [Restoring an individual element from backup . .](#_bookmark157) [161](#_bookmark157) [Restoring a view from backup](#_bookmark158) [162](#_bookmark158)

[Synchronizing VOBs and views after you restore](#_bookmark159) [163](#_bookmark159)

[To find views with references in a VOB . . .](#_bookmark159) [163](#_bookmark159) [To synchronize dynamic views](#_bookmark160) [164](#_bookmark160)

[To reconcile missing checkouts](#_bookmark160) [164](#_bookmark160)

[To remove eclipsing view-private files](#_bookmark160) [164](#_bookmark160)

[Remove winked-in DOs that are not valid . . .](#_bookmark161) [165](#_bookmark161) [Reestablish the consistency of a dynamic view’s](#_bookmark161) [derived object state](#_bookmark161) [165](#_bookmark161)

[Restoring one or more members of a set of related](#_bookmark161) [databases](#_bookmark161) [165](#_bookmark161)

[Restoring a component VOB](#_bookmark162) [166](#_bookmark162)

[Restoring a Rational ClearQuest database . . .](#_bookmark162) [166](#_bookmark162) [Special procedures for restoring a PVOB . . .](#_bookmark163) [167](#_bookmark163) [Restoring a member of an administrative VOB](#_bookmark163) [hierarchy](#_bookmark163) [167](#_bookmark163)

#### [Chapter 12. Periodic maintenance . .](#_bookmark164) [169](#_bookmark164)

[The Rational ClearCase scheduler](#_bookmark164) [169](#_bookmark164)

[Managing the scheduler access control list. . .](#_bookmark164) [169](#_bookmark164)

[Creating a task](#_bookmark166) [171](#_bookmark166)

[Editing a task](#_bookmark166) [171](#_bookmark166)

[Deleting a task](#_bookmark167) [172](#_bookmark167)

[Managing jobs](#_bookmark167) [172](#_bookmark167)

[Creating a job](#_bookmark167) [172](#_bookmark167)

[Specifying a job’s schedule](#_bookmark167) [172](#_bookmark167)

[Specifying job notifications](#_bookmark168) [173](#_bookmark168)

[Viewing job properties](#_bookmark169) [174](#_bookmark169)

[Editing job properties](#_bookmark169) [174](#_bookmark169)

[Running a job immediately](#_bookmark170) [175](#_bookmark170)

[Deleting a job](#_bookmark170) [175](#_bookmark170)

[Scrubbing to control VOB storage growth](#_bookmark171) [176](#_bookmark171)

[Scrubbing VOB storage pools](#_bookmark171) [176](#_bookmark171)

[Scrubbing VOB databases](#_bookmark171) [176](#_bookmark171)

[Adjusting default scrubbing parameters . . .](#_bookmark172) [177](#_bookmark172) [Scrubbing derived objects more often . . .](#_bookmark172) [177](#_bookmark172) [Fine-tuning derived object scrubbing . . .](#_bookmark172) [177](#_bookmark172) [Scrubbing less aggressively](#_bookmark173) [178](#_bookmark173)

[About checkvob](#_bookmark174) [179](#_bookmark174)

[Replicated VOB considerations](#_bookmark174) [179](#_bookmark174)

[Using checkvob to find and fix internal VOB](#_bookmark175) [inconsistencies](#_bookmark175) [180](#_bookmark175)

[Individual file element or DO processing . . .](#_bookmark175) [180](#_bookmark175) [Pool mode processing](#_bookmark176) [181](#_bookmark176)

[Descriptions of storage pool problems](#_bookmark176) [181](#_bookmark176)

[Pool name and protection problems](#_bookmark176) [181](#_bookmark176)

[Source, DO, or cleartext pool: bad pool roots](#_bookmark177) [182](#_bookmark177)

[Source or DO pool: misprotected container](#_bookmark177)

[on Windows](#_bookmark177) [182](#_bookmark177)

[Missing and unreferenced data containers . . .](#_bookmark177) [182](#_bookmark177) [Source pool: missing container](#_bookmark178) [183](#_bookmark178)

[Source Pool: Unreferenced container (debris)](#_bookmark178) [183](#_bookmark178)

[DO pool: Missing container](#_bookmark179) [184](#_bookmark179)

[DO pool: Unreferenced container (debris)](#_bookmark179) [184](#_bookmark179)

[Minimizing data loss with checkvob -force -fix](#_bookmark179) [184](#_bookmark179)

[Using checkvob to find and fix problems with](#_bookmark180)

[global types](#_bookmark180) [186](#_bookmark180)

[Using checkvob to find and fix broken hyperlinks](#_bookmark181) [187](#_bookmark181)

[Using checkvob in a UCM environment](#_bookmark182) [188](#_bookmark182)

[View storage maintenance for dynamic views . .](#_bookmark182) [188](#_bookmark182) [Getting information on view contents](#_bookmark182) [188](#_bookmark182)

[Scrubbing view-private storage](#_bookmark183) [189](#_bookmark183)

[Cleaning up a view manually](#_bookmark183) [189](#_bookmark183)

[Chapter 13. Importing data](#_bookmark184) [191](#_bookmark184)

[Importing data from Microsoft Visual SourceSafe](#_bookmark184) [191](#_bookmark184)

[Overview of the example Microsoft Visual](#_bookmark184) [SourceSafe configuration](#_bookmark184) [191](#_bookmark184)

[Setting your environment](#_bookmark186) [193](#_bookmark186)

[Running clearexport\_ssafe](#_bookmark186) [193](#_bookmark186)

[Running clearimport](#_bookmark187) [194](#_bookmark187)

[Examining the results](#_bookmark187) [194](#_bookmark187)

[Importing RCS data](#_bookmark188) [195](#_bookmark188)

[Creating the data file](#_bookmark188) [195](#_bookmark188)

[Running clearimport](#_bookmark189) [196](#_bookmark189)

[Importing PVCS data](#_bookmark189) [196](#_bookmark189)

[Creating the data file](#_bookmark189) [196](#_bookmark189)

[Running the conversion scripts](#_bookmark190) [197](#_bookmark190)

[Chapter 14. Troubleshooting](#_bookmark191) [199](#_bookmark191)

[Repairing storage directory ACLS on NTFS . . .](#_bookmark191) [199](#_bookmark191) [VOB and view storage directory ACLs](#_bookmark191) [199](#_bookmark191)

[Preserving NTFS ACLs when copying a VOB or](#_bookmark192) [view storage directory](#_bookmark192) [200](#_bookmark192)

[Causes of protection problems](#_bookmark193) [201](#_bookmark193)

[Copying the storage directory incorrectly . .](#_bookmark193) [201](#_bookmark193) [Converting the file system to NTFS](#_bookmark193) [201](#_bookmark193)

[Editing permissions](#_bookmark193) [201](#_bookmark193)

[Utilities for fixing protection problems](#_bookmark193) [201](#_bookmark193)

[fix\_prot](#_bookmark194) [202](#_bookmark194)

[lsacl](#_bookmark195) [203](#_bookmark195)

[Fixing protection problems](#_bookmark195) [203](#_bookmark195)

[Problems with existing views after relocating](#_bookmark196)

[elements](#_bookmark196) [204](#_bookmark196)

[Cleanup guidelines](#_bookmark196) [204](#_bookmark196)

[Problems with symbolic links to relocated](#_bookmark197)

[elements](#_bookmark197) [205](#_bookmark197)

[Updating directory versions manually](#_bookmark198) [206](#_bookmark198)

[Fixing symbolic links created by relocate . .](#_bookmark198) [206](#_bookmark198) [Modifying old target directory versions to](#_bookmark198)

[see relocated elements](#_bookmark198) [206](#_bookmark198)

[Modifying newest version of source directory](#_bookmark199)

[to see relocated elements](#_bookmark199) [207](#_bookmark199)

[Locating MVFS data containers](#_bookmark199) [207](#_bookmark199)

[Scenario](#_bookmark199) [207](#_bookmark199)

[Determining the status of Rational ClearCase](#_bookmark199)

[files](#_bookmark199) [207](#_bookmark199)

[Determining the full pathnames of files . . .](#_bookmark199) [207](#_bookmark199) [Where is the VOB?](#_bookmark200) [208](#_bookmark200)

[Where is the view?](#_bookmark200) [208](#_bookmark200)

[Where are the individual files?](#_bookmark200) [208](#_bookmark200)

[Locating a checked-out version](#_bookmark201) [209](#_bookmark201)

[Locating a checked-in version’s cleartext](#_bookmark201)

[container](#_bookmark201) [209](#_bookmark201)

[Locating a checked-in version’s source](#_bookmark201)

[container](#_bookmark201) [209](#_bookmark201)

[Locating a view-private file](#_bookmark201) [209](#_bookmark201)

[Issues with remote pools on Linux and the](#_bookmark201)

[UNIX system](#_bookmark201) [209](#_bookmark201)

[Links and directories on Linux and the UNIX](#_bookmark202) [system](#_bookmark202) [210](#_bookmark202)

[Preventing recursive traversal of the view root](#_bookmark202)

[directory on Linux or the UNIX system](#_bookmark202) [210](#_bookmark202)

[Chapter 15. Improving client host](#_bookmark203) [performance](#_bookmark203) [211](#_bookmark203)

[Client host configuration guidelines](#_bookmark203) [211](#_bookmark203)

[Examining and adjusting MVFS cache size . .](#_bookmark203) . [211](#_bookmark203) [Examining MVFS cache statistics](#_bookmark204) [212](#_bookmark204)

[Adjusting the MVFS scaling factor](#_bookmark205) [213](#_bookmark205)

[Setting individual caching parameters](#_bookmark206) [214](#_bookmark206)

[Minimizing attribute cache misses](#_bookmark207) [216](#_bookmark207)

[Attribute cache total misses](#_bookmark207) [216](#_bookmark207)

[Close-to-Open misses](#_bookmark207) [216](#_bookmark207)

[Generation misses](#_bookmark207) [216](#_bookmark207)

[Cache time-out misses](#_bookmark207) [216](#_bookmark207)

[Cache fill misses](#_bookmark208) [217](#_bookmark208)

[Event time misses](#_bookmark208) [217](#_bookmark208)

[View caches](#_bookmark208) [217](#_bookmark208)

[Obtaining view cache information](#_bookmark209) [218](#_bookmark209)

[Analyzing the output](#_bookmark209) [218](#_bookmark209)

[Reconfiguring a view](#_bookmark210) [219](#_bookmark210)

[Reference information](#_bookmark210) [219](#_bookmark210)

[mvfscache](#_bookmark210) [219](#_bookmark210)

[mvfsstat](#_bookmark211) [221](#_bookmark211)

#### [Chapter 16. Improving VOB host](#_bookmark212) [performance](#_bookmark212) [225](#_bookmark212)

[Minimize process overhead](#_bookmark212) [225](#_bookmark212)

[Maximize disk performance](#_bookmark212) [225](#_bookmark212)

[Add memory for disk or file system caching . . .](#_bookmark213) [226](#_bookmark213) [Modify lock manager startup options](#_bookmark213) [226](#_bookmark213)

[Lock manager implementations](#_bookmark213) [226](#_bookmark213)

[To change lock manager startup options . . .](#_bookmark213) [226](#_bookmark213) [On Windows hosts](#_bookmark213) [226](#_bookmark213)

[On hosts running Linux or the UNIX system](#_bookmark214) [227](#_bookmark214)

[lockmgr reference information](#_bookmark214) [227](#_bookmark214)

[Server performance and NAS devices](#_bookmark215) [229](#_bookmark215)

#### [Appendix A. Configuring](#_bookmark216)

#### [cross-platform file system access . .](#_bookmark216) [231](#_bookmark216)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [NFS client products . . . . . . .](#_bookmark216) | [.](#_bookmark216) | [.](#_bookmark216) | [.](#_bookmark216) | [.](#_bookmark216) | [231](#_bookmark216) |
| [Disabling automatic case conversion.](#_bookmark216) | [.](#_bookmark216) | [.](#_bookmark216) | [.](#_bookmark216) | [.](#_bookmark216) | [231](#_bookmark216) |
| [Shaffer Solutions DiskAccess . .](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [232](#_bookmark217) |
| [Hummingbird NFS Maestro . .](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [232](#_bookmark217) |

[Setting an NFS client’s default protection . . .](#_bookmark217) [232](#_bookmark217) [Shaffer Solutions DiskAccess](#_bookmark217) [232](#_bookmark217)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [Hummingbird NFS Maestro](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [.](#_bookmark217) | [232](#_bookmark217) |
| [Setting the correct logon name](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [233](#_bookmark218) |
| [Shaffer Solutions DiskAccess](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [233](#_bookmark218) |
| [Hummingbird NFS Maestro](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [.](#_bookmark218) | [233](#_bookmark218) |

[Accessing the Syntax Administration](#_bookmark226)

[Framework](#_bookmark226) [241](#_bookmark226)

[Performing initial setup of TAS](#_bookmark226) [241](#_bookmark226)

[General TAS settings](#_bookmark227) [242](#_bookmark227)

[Enabling and configuring the CIFS realm](#_bookmark227)

[(TAS 6.x)](#_bookmark227) [242](#_bookmark227)

[Mapping the Rational ClearCase server](#_bookmark227)

[process user (TAS 6.x)](#_bookmark227) [242](#_bookmark227)

[Creating a volume (TAS 6.x)](#_bookmark228) [243](#_bookmark228)

[Configuring the file service (TAS 6.x) . . .](#_bookmark228) [243](#_bookmark228) [Configuring TAS 7.x to support Rational](#_bookmark229) [ClearCase . . . . . . . . . . .](#_bookmark229)

|  |  |
| --- | --- |
| [.](#_bookmark229) | [244](#_bookmark229) |
|  | [246](#_bookmark230) |
| [.](#_bookmark230) | [246](#_bookmark230) |
| [.](#_bookmark230) | [246](#_bookmark230) |
| [.](#_bookmark230) | [246](#_bookmark230) |

[Start services and accept service connections](#_bookmark230) [Configuring Rational ClearCase to support](#_bookmark230) [TAS . . . . . . . . . . . . .](#_bookmark230)

[Testing the TAS configuration on ordinary](#_bookmark230) [files . . . . . . . . . . . . .](#_bookmark230)

[Testing the TAS configuration with Rational](#_bookmark230) [ClearCase . . . . . . . . . . .](#_bookmark230)

#### [Appendix B. Rational ClearCase and](#_bookmark231) [Windows domains](#_bookmark231) [249](#_bookmark231)

[Domain configurations compatible with Rational](#_bookmark231) [ClearCase](#_bookmark231) [249](#_bookmark231)

[What Rational ClearCase requires from any](#_bookmark231)

[domain](#_bookmark231) [249](#_bookmark231)

[Rational ClearCase on nondomain hosts . . .](#_bookmark231) [249](#_bookmark231) [Domain user and group accounts](#_bookmark232) [250](#_bookmark232)

|  |  |  |
| --- | --- | --- |
| [Setting the Rational ClearCase primary group](#_bookmark233) |  | [251](#_bookmark233) |
| [Defining required domain accounts manually](#_bookmark233) |  | [251](#_bookmark233) |
| [Multiple user account domain support . . . .](#_bookmark234) | [.](#_bookmark234) | [252](#_bookmark234) |
| [Using active directory universal groups . .](#_bookmark234) | [.](#_bookmark234) | [252](#_bookmark234) |
| [Using proxy groups and domain mapping in](#_bookmark235) |  |  |

[server process user . . . . . . . .](#_bookmark219) [Saving the user identity in a Windows](#_bookmark220) [registry file . . . . . . . . . .](#_bookmark220)

|  |  |  |
| --- | --- | --- |
| [Hummingbird NFS Maestro: disabling MS-DOS](#_bookmark218) |  | |
| [sharing . . . . . . . . . . . . .](#_bookmark218) | [.](#_bookmark218) | [233](#_bookmark218) |
| [Automounting and NFS client software . .](#_bookmark219) | [.](#_bookmark219) | [234](#_bookmark219) |
| [DiskAccess: supporting the Rational ClearCase](#_bookmark219) |  |  |

|  |  |  |
| --- | --- | --- |
| [Windows tags for VOBs on Linux or the UNIX](#_bookmark220) |  | |
| [system with remote storage pools . . . .](#_bookmark220) | [.](#_bookmark220) | [235](#_bookmark220) |
| [Poolmap syntax . . . . . . . . .](#_bookmark221) | [.](#_bookmark221) | [236](#_bookmark221) |
| [Mapping storage pools in an existing VOB](#_bookmark222) |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [Setting VOB storage ACLs .](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [254](#_bookmark236) |
| [Conversion to Active Directory .](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [254](#_bookmark236) |
| [Understanding Active Directory](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [254](#_bookmark236) |

[How Active Directory affects Rational ClearCase](#_bookmark236) [254](#_bookmark236) [Planning your Active Directory upgrade or](#_bookmark237)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [Windows NT domains . . . . .](#_bookmark235) | [.](#_bookmark235) | [.](#_bookmark235) | [.](#_bookmark235) | [.](#_bookmark235) | [253](#_bookmark235) |
| [Setting VOB element permissions.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [.](#_bookmark236) | [254](#_bookmark236) |

|  |  |  |
| --- | --- | --- |
| [.](#_bookmark219) | [.](#_bookmark219) | [234](#_bookmark219) |
| [.](#_bookmark220) | [.](#_bookmark220) | [235](#_bookmark220) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [migration strategy . . . . . . .](#_bookmark237) | [.](#_bookmark237) | [.](#_bookmark237) | [.](#_bookmark237) | [.](#_bookmark237) | [255](#_bookmark237) |
| [Preparing Rational ClearCase hosts .](#_bookmark237) | [.](#_bookmark237) | [.](#_bookmark237) | [.](#_bookmark237) | [.](#_bookmark237) | [255](#_bookmark237) |
| [Domain upgrade scenarios . . . . .](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [256](#_bookmark238) |
| [Upgrading a single domain. . . .](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [256](#_bookmark238) |

[Mapping the Rational ClearCase server](#_bookmark222)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [tag . . . . . . . . . .](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [237](#_bookmark222) |
| [SMB server products . . . . . .](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [237](#_bookmark222) |
| [Installing and configuring Samba.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [.](#_bookmark222) | [237](#_bookmark222) |

[process user](#_bookmark222) [237](#_bookmark222)

[Using the Samba Web Administration Tool](#_bookmark223)

[Upgrading a master domain and its resource](#_bookmark238) [domains](#_bookmark238) [256](#_bookmark238)

[Upgrading multiple master and resource](#_bookmark238)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [domains . . . . . . .](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [.](#_bookmark238) | [256](#_bookmark238) |
| [Converting proxy groups](#_bookmark239) | [.](#_bookmark239) | [.](#_bookmark239) | [.](#_bookmark239) | [.](#_bookmark239) | [.](#_bookmark239) | [.](#_bookmark239) | [.](#_bookmark239) | [257](#_bookmark239) |
| [Domain migration scenarios .](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [258](#_bookmark240) |
| [Migrating multiple domains](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [.](#_bookmark240) | [258](#_bookmark240) |

[Starting Samba services](#_bookmark224) [239](#_bookmark224)

|  |  |  |
| --- | --- | --- |
| [(SWAT) . . . . . . . . . . . .](#_bookmark223) | [.](#_bookmark223) | [238](#_bookmark223) |
| [Configuring Samba globals for Rational](#_bookmark223) |  |  |
| [ClearCase . . . . . . . . . . .](#_bookmark223) | [.](#_bookmark223) | [238](#_bookmark223) |
| [Creating shares for VOB and view storage](#_bookmark224) |  | [239](#_bookmark224) |

[Configuring Rational ClearCase to support](#_bookmark225)

[Samba](#_bookmark225) [240](#_bookmark225)

[Testing the Samba configuration on](#_bookmark225)

[non-ClearCase files](#_bookmark225) [240](#_bookmark225)

[Testing the Samba configuration with](#_bookmark225)

[Rational ClearCase](#_bookmark225) [240](#_bookmark225)

[Installing and configuring TAS](#_bookmark226) [241](#_bookmark226)

[Enabling the multiuser kernel driver on AIX](#_bookmark226) [241](#_bookmark226)

[Migrating users and groups](#_bookmark240) [258](#_bookmark240)

[Using vob\_sidwalk to change or update VOB users](#_bookmark243)

|  |  |  |
| --- | --- | --- |
| [If you must add a new user while migration](#_bookmark241) |  | |
| [is in progress . . . . . . . . . .](#_bookmark241) | [.](#_bookmark241) | [259](#_bookmark241) |
| [Migrating individual hosts . . . . . .](#_bookmark241) | [.](#_bookmark241) | [259](#_bookmark241) |
| [If VOB servers cannot migrate when clients](#_bookmark242) |  |  |
| [do . . . . . . . . . . . . . .](#_bookmark242) | [.](#_bookmark242) | [260](#_bookmark242) |

[and groups](#_bookmark243) [261](#_bookmark243)

[Remapping historical SIDs after domain](#_bookmark244)

[migration](#_bookmark244) [262](#_bookmark244)

[Remapping current SIDs when moving a VOB](#_bookmark244)

[to a new domain](#_bookmark244) [262](#_bookmark244)

[Reassigning ownership to the VOB owner. . .](#_bookmark244) [262](#_bookmark244) [Resetting VOB storage directory protections . .](#_bookmark245) [263](#_bookmark245) [Using -delete\_groups with replicas that preserve](#_bookmark245) [identities and permissions](#_bookmark245) [263](#_bookmark245)

#### [Appendix C. Configuring](#_bookmark246)

#### [non-ClearCase access on Linux or the](#_bookmark246) [UNIX system](#_bookmark246) [265](#_bookmark246)

[Using non-ClearCase access on hosts running](#_bookmark246)

[Linux or the UNIX system](#_bookmark246) [265](#_bookmark246)

[Restrictions on use](#_bookmark246) [265](#_bookmark246)

[Setting up an export view for Non-ClearCase](#_bookmark246)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [access . . . . . . . .](#_bookmark246) | [.](#_bookmark246) | [.](#_bookmark246) | [.](#_bookmark246) | [.](#_bookmark246) | [.](#_bookmark246) | [.](#_bookmark246) | [.](#_bookmark246) | [.](#_bookmark246) | [265](#_bookmark246) |
| [Exporting multiple VOBs](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [266](#_bookmark247) |
| [Export configurations. .](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [.](#_bookmark247) | [266](#_bookmark247) |

[Avoiding multihop configurations](#_bookmark248) [267](#_bookmark248)

[Working with multihop configurations . . .](#_bookmark248) [267](#_bookmark248) [Restricting exports to particular hosts . . .](#_bookmark248) [267](#_bookmark248) [Using automount with non-ClearCase access . . .](#_bookmark248) [267](#_bookmark248) [NFS problems with Non-ClearCase access](#_bookmark249) [268](#_bookmark249)

[Problems with NFS client caching](#_bookmark249) [268](#_bookmark249)

[Problems with NFS locking](#_bookmark250) [269](#_bookmark250)

#### [Appendix D. Estimating VOB size. . .](#_bookmark251) [271](#_bookmark251)

[Estimating database capacity](#_bookmark251) [271](#_bookmark251)

[Estimating pool capacity](#_bookmark252) [272](#_bookmark252)

# Figures

1. [Registry with one region](#_bookmark34) [29](#_bookmark34)
2. [Registry with two regions](#_bookmark35) [30](#_bookmark35)
3. [How cleartool commands affect registry data](#_bookmark40) [35](#_bookmark40)
4. [Administrative VOB hierarchy](#_bookmark97) [97](#_bookmark97)
5. [Replication requirements of administrative](#_bookmark107)

[VOB hierarchies](#_bookmark107) [107](#_bookmark107)

1. [Database relationships](#_bookmark149) [153](#_bookmark149)
2. [Sample Microsoft Visual SourceSafe](#_bookmark185)

[configuration](#_bookmark185) [192](#_bookmark185)

1. [Version tree of imported element](#_bookmark188) [195](#_bookmark188)

# Tables

* 1. [Characters not allowed in Windows file names](#_bookmark56) [53](#_bookmark56)
  2. [Protection mode for a Rational ClearCase](#_bookmark60)

[object](#_bookmark60) [58](#_bookmark60)

* 1. [Protection mode digits for a Rational](#_bookmark60)

[ClearCase object](#_bookmark60) [58](#_bookmark60)

* 1. [Importance of VOB directories in partial](#_bookmark145)

[backups](#_bookmark145) [149](#_bookmark145)

* 1. [Access types in scheduler ACL entries](#_bookmark165) [170](#_bookmark165)
  2. [Default scrubber and vob\_scrubber actions](#_bookmark171) [176](#_bookmark171)
  3. [Storage directories for MVFS files](#_bookmark200) [208](#_bookmark200)
  4. [How memory size affects the MVFS scaling](#_bookmark205)

[factor](#_bookmark205) [213](#_bookmark205)

* 1. [How the MVFS scaling factor affects](#_bookmark205)

[individual MVFS cache sizes](#_bookmark205) [213](#_bookmark205)

* 1. [MVFS cache information (Part 1 of 2)](#_bookmark206) [214](#_bookmark206)
  2. [Samba global settings for Rational ClearCase](#_bookmark224) [239](#_bookmark224)
  3. [Sizes of common VOB data types](#_bookmark251) [271](#_bookmark251)
  4. [Type manager compression ratios](#_bookmark252) [272](#_bookmark252)
  5. [Files containing default RWP port](#_bookmark255)

[information](#_bookmark255) [277](#_bookmark255)

* 1. [RWP log levels](#_bookmark256) [278](#_bookmark256)
  2. [Configuration parameters in ccweb.conf](#_bookmark264) [290](#_bookmark264)
  3. [Rational ClearCase Web server configuration](#_bookmark266) [steps according to Rational ClearCase and](#_bookmark266)

[Rational ClearQuest functionality](#_bookmark266) [292](#_bookmark266)

# About this book

This manual is for users or administrators of IBM Rational ClearCase and IBM 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 Rational ClearCase and Rational ClearCase LT on all supported platforms:

v Managing client and server hosts.

v Administering licenses.

v Administering the Rational ClearCase registry, a central directory of shared information.

v Managing user and group accounts.

v Creating and managing VOBs (versioned object bases, or data repositories).

v Creating and managing views (user workspaces).

v Importing data from other configuration management systems.

v Periodic maintenance and troubleshooting.

v Tuning client and server hosts for better performance.

v Configuring optional vendor access software.

v Configuring the Rational® Web Platform, the Rational ClearCase Web interface, and the Rational ClearCase Remote Client.

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

# Chapter 4. Rational ClearCase network administration in mixed environments

Whenever a Rational ClearCase community

v Common user and group names and common primary group memberships on each platform for all users and groups that access a common set of VOBs and views

v 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 computer running Linux or the UNIX system

v Client configuration issues and file naming conventions that address differences among the supported operating systems 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 Rational ClearCase community whose computers all run any of these operating systems, you can skip this chapter.

## Common user and group names

The identity with which a user logs on to the operating system of a Rational ClearCase host establishes the credentials that determine the access rights to artifacts in VOBs and views.

Because access rights are associated with users and groups by name, each user who accesses a VOB or dynamic view from computers running supported operating systems must have the same user name on all of them. In addition, all groups whose members access VOBs and views must have the same names on all supported operating systems, and each user’s primary group assignment must be the same in all environments. [Chapter 5](#_bookmark57) describes how credentials are used when determining access rights in VOBs and views.

When a user’s credentials are evaluated by Rational ClearCase, group membership is considered only when the user requests access to file system objects such as directory and file elements. For access to VOB metadata such as labels and branches, only the owner’s name is important. Group membership is not considered.

## How user and group names are compared by Rational ClearCase

The user and group name comparisons made by Rational ClearCase are not

case-sensitive on Windows and case-sensitive on Linux and on the UNIX system. 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:** Supported operating systems 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 Rational ClearCase community must be acceptable in both environments.

## Credentials mapping

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

The names in the credentials are compared and, if they can be matched, the combined *user.group* credentials are evaluated on the other operating system 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 Linux or on the UNIX system, 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 Rational ClearCase server host.

### Specifying the Windows domain for credential mapping of users on hosts running Linux or the UNIX system

If your VOB, view, registry, or license server hosts will be accessed by clients running Linux or the UNIX system, you must specify the Windows domain that will be used to map the credentials of those users.

Specify the Windows domain for credential mapping:

###### Click Start > Settings > Control Panel.

1. Double-click**ClearCase** and click the **Options** page.
2. In the **Use this domain to map UNIX user and group names** area, select a Windows domain in which user and group identities have been defined for all Linux or UNIX system-based users who access these services on this server.

### Credentials for the Rational ClearCase server process user on Linux and the UNIX system

The Rational ClearCase server process user account is a special Windows domain account (see [“Domain user and group accounts” on page 250)](#_bookmark232) under whose identity the **albd\_server** runs on Windows computers. Mixed environments in which dynamic views on Windows access VOBs on Linux or on the UNIX system require a user account to which the Rational ClearCase server process user’s credentials can be mapped. (No server processes on Linux or the UNIX system run with the identity of this account. It is used only for credentials mapping when the Rational ClearCase server process user requests access to a VOB or dynamic view on Linux or the UNIX system.)

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

v The account must be a member of each group that owns a VOB or dynamic view hosted on a computer running Linux or the UNIX system that will be accessed from Windows.

v The account’s primary group should have the same name as the primary group designated for Windows Rational ClearCase users. (If it does not already exist, you must create this group on the system running Linux or the UNIX system.)

**Note:** On Windows computers, the Rational ClearCase server process user is a member of a single group, the Rational ClearCase administrators group, that has special Rational ClearCase privileges but no special privileges in the operating system. Even though some Rational ClearCase server processes run as **root** on Linux and on the UNIX system, the Rational ClearCase server process user account should not be mapped to **root**.

Privileged user status is not required for this account, and mapping it to

**root** might create a security risk.

v 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 Linux and the UNIX system; that NFS client does not allow you to map the Rational ClearCase server process user account to a Linux or UNIX system-based account with a different name. (For more information, see [“NFS client](#_bookmark216) [products” on page 231.)](#_bookmark216)
* VOB and view server hosts running Linux and the UNIX system run an SMB server to provide Windows computers with access to VOBs and views, and you do not want to map the Rational ClearCase server process user account to an existing account on Linux or the UNIX system. (For more information, see [“SMB server products” on page](#_bookmark222) 237.)

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

## Cross-platform file system access

Dynamic views must use a network file system to access artifacts such as constructed versions and derived objects in VOB storage directories or files in remote view storage. When the view and VOB are on the same platform type, its native network file system is used. But when they are on different platform types, cross-platform file system access is required.

If you need cross-platform file system access, you must install and configure vendor software to provide it. For more information, see [Appendix A,](#_bookmark216) [“Configuring cross-platform file system access,” on page 231.](#_bookmark216) If a Rational ClearCase client uses only snapshot or Web views, no vendor software

cross-platform file-access solution is needed to access VOB data.

Cross-platform file system access is supported only to allow dynamic views on Windows computers to access files in VOBs and views on Linux or the UNIX system. IBM does not support any cross-platform file system access product that enables dynamic views on computers running Linux or the UNIX system to access files in VOBs and views on Windows.

## The Rational ClearCase File Service

The Rational ClearCase File Service is a TCP/IP-based mechanism that is included in both Rational ClearCase and Rational ClearCase LT. When CCFS is in use, file transfers between a VOB and snapshot view (for example, operations such as adding a file to source control, updating it in the view, and checking it out or in) take place over a standard TCP/IP connection created by CCFS.

CCFS is used whenever a snapshot view accesses VOB data. CCFS is used by Rational ClearCase to enable cross-platform file transfers between VOBs and snapshot views.

v On Linux and on the UNIX system, CCFS is used to support access by snapshot views to VOBs on Windows.

v On Windows, the native network file system is normally used to support access to by snapshot views to VOBs on Windows. You have the option of enabling CCFS to support access by snapshot views to VOBs on Linux and on the UNIX system.

### Enabling CCFS on Windows hosts

To enable CCFS on a Rational ClearCase Windows computer:

1. Click **Start > Settings > Control Panel**, and then start the ClearCase program.
2. On the **Options** page, select the **Use CCFS to access UNIX VOBs** check box to enable 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 VOBs on Linux and the UNIX system. 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 VOBs hosted on Linux or the UNIX system cannot provide **view\_server** support for Rational ClearCase clients that cannot run a local **view\_server** process.

## Network-attached storage and cross-platform file access tools

IBM 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 computers running any supported operating system, without the need to install any other software on Rational ClearCase client or server hosts. For more information, see [“Rational ClearCase and](#_bookmark17)

[network-attached storage devices” on page 12.](#_bookmark17)

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

v SMB servers for platforms running Linux or the UNIX system, which provide Windows clients with native SMB/CIFS access to file systems on hosts running Linux or the UNIX system.

v NFS client programs for Windows, which enable Windows clients to use the NFS protocol to access the file systems of computers running Linux or the UNIX system.

Any of these tools can be used to enable a dynamic view on a Windows computer to access a VOB hosted on a computer running Linux or the UNIX system and to enable a Windows computer to use shared dynamic views hosted on computers running Linux or the UNIX system.

For more information about supported cross-platform file system access tools, see [Appendix A, “Configuring cross-platform file system access,” on page 231.](#_bookmark216)

## Case-sensitivity

Linux, the UNIX system, and Windows observe different conventions for

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

v Regardless of how you configure Rational ClearCase, all Rational ClearCase applications that refer to pathnames in VOBs, including **cleartool** and **clearmake**, are case-sensitive.

v The MVFS on Linux and on the UNIX system 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 Rational ClearCase on all supported operating systems 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:

v In VOBs and views, avoid creating file or directory names that differ only in character capitalization. An MVFS that is not case-sensitive cannot distinguish between two file names that differ only in their capitalization. Attempts to access such files produce indeterminate results.

v Configure the MVFS on Windows to be not case-sensitive (this is the default setting).

v If you use an NFS client or SMB server product, disable any automatic case conversion features it may have (See [Appendix A, “Configuring cross-platform](#_bookmark216) [file system access,” on page 231).](#_bookmark216)

The remainder of this section explains pertinent case-sensitivity issues, and discusses each recommendation in more detail.

## Case-sensitivity and the MVFS

The multiversion file system (MVFS) is a Rational ClearCase feature that supports dynamic views. Because it is a file system, its case-sensitivity behavior is important and needs to be reconfigured in some mixed networks. For more information, see [“The Multiversion File System” on page 133.](#_bookmark130)

**Note:** Rational ClearCase LT does not support the MVFS.

On Linux and on the UNIX system, the native file system and all Rational 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-preserving and not case-sensitive. Because native Windows file systems perform file lookup that is not case-sensitive, the MVFS, in its default configuration on Windows, also performs file lookup that is not case-sensitive. 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** page in the ClearCase program in Control Panel:

v 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.

v 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 might 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 an MVFS environment that is not case-sensitive, 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 on which the snapshot view directory is located to read and write files. On Windows, these file systems perform file lookup that is not case-sensitive and that generally preserves case when creating file names. On Linux and the UNIX system, file lookups are always case-sensitive, and file-name creation is always case-preserving.

## Configuring text modes for views

The operating systems that Rational ClearCase supports observe different conventions for writing line terminations in text files. Utilities and applications on Linux and the UNIX system 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 computers running different supported operating systems. For example, a file that contains

abc def ghi

Would look like this if it were created by a Windows editor such as Notepad and read by a Linux or UNIX system-based editor such as **vi**:

abc^M def^M ghi^M

The text editor on the Linux or UNIX system renders the <CR> character as ^M. The same file would look like this if it were created by the system editor on Linux or the UNIX system and read by the Windows editor:

abc"def"ghi

To better support parallel development in mixed environments, a text mode setting is provided 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 Rational 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:

v **transparent text mode**. In a view created in **transparent** text mode, no

line-terminator processing is performed by Rational ClearCase. 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 or use tools that are compatible with either line-termination convention, all views should be created in **transparent** text mode.

v **insert\_cr text mode.** In a view created in **insert\_cr** text mode, a <CR> character is inserted 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.

v **strip\_cr text mode.** In a view created in **strip\_cr** text mode, the <CR> character is stripped 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, the <CR> characters are added or removed when the view is updated. In a dynamic view, the <CR> characters are added or removed when you open and read files. For both snapshot views and dynamic views, the <CR> manipulation is reversed (adding or removing <CR> characters as appropriate) during the checkin process.

### Determining the text mode for a view

If you do not know the text mode for a view, you can find out in one of the following ways:

v In Windows Explorer, right-click a drive that represents a view. Then click **ClearCase > Properties of View**. The text mode is displayed on the **Access** page.

v 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

No policy governing access to VOBs is enforced by Rational ClearCase 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 supported 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 Linux or the UNIX system:

v Clients running Linux or the UNIX system should use views created in

**transparent** text mode.

v Windows clients should use views created in **insert\_cr** text mode.

If most users are editing text files on Windows:

v Windows clients should use views created in **transparent** text mode.

v Clients running Linux or the UNIX system 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 Linux or the UNIX system creates a version of an element in a view that has a **strip\_cr** or **insert\_cr** text mode and another user 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 in earlier versions of Rational 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:

v 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.

v 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 Rational ClearCase MultiSite users

The text mode characteristics supported by a VOB are not replicated by Rational ClearCase MultiSite. 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 file names on Linux and the UNIX system 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 1](#_bookmark56) lists these characters.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Table 1. Characters not allowed in Windows file names* | | | | | | |
| ? | \* | / | \ | | | < | > |

# Chapter 5. VOB and view access control

## Fundamentals of VOB and view access control

Access controls implemented within Rational ClearCase 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 Rational ClearCase objects, and the credentials of user processes or application programs that access Rational ClearCase data on behalf of users.

## Users and groups

Implementation of user and group accounts are not handled in a special way within Rational ClearCase. Instead, the identity with which a user logs on to the operating system of a Rational ClearCase host establishes the user’s Rational ClearCase credentials. Because these credentials are evaluated whenever and wherever a user requests access to an object under Rational ClearCase control, operating system definitions for user names, group names, and each user’s group memberships must be consistent on every Rational ClearCase host. This consistency is usually achieved by means of an account database such as a Windows domain or the Network Information System (NIS) supported on Linux and the UNIX system.

**Note:** In environments where users access a common set of VOBs and views from hosts running different supported operating systems, 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 Linux and the UNIX system as they are on Windows). For more information, see [“Common user and group](#_bookmark48) [names” on page 45.](#_bookmark48)

### 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. Make all Rational ClearCase users in a community members of the same primary group. This can be a group that already exists, or one that you create expressly for use by the Rational ClearCase community.

The primary group is defined in a different way on each supported operating system:

v On Linux and on the UNIX system, the primary group is defined in the user’s entry in the NIS **passwd** database.

v 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 Rational 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 Rational ClearCase users on Windows (see [“Setting the Rational ClearCase](#_bookmark233) [primary group” on page 251).](#_bookmark233)

### Limitations when a user belongs to more than 32 groups

If a user is a member of more than 32 groups, only the first 32 groups (in numerical order by GID on Linux and on the UNIX system, or SID on Windows) are recognized by Rational ClearCase. If the user environment variable CLEARCASE\_GROUPS exists for any user, the semicolon-separated list of group names specified in the value of this variable first will be considered by Rational ClearCase when determining the list of groups to which the user belongs.

## Privileged users and groups

A typical Rational ClearCase community includes two classes of user:

v **Ordinary users** have rights to modify or delete Rational 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.

v **Privileged users** have unrestricted rights to create, modify, and delete all Rational ClearCase resources.

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

v On a Rational ClearCase or Rational ClearCase LT server running Linux or the UNIX system, 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 view access privileges for remote root” on page](#_bookmark58) 56.)

v On a Rational ClearCase Windows host, the privileged user is any member of the Rational ClearCase administrators group, described in [“Domain user and](#_bookmark232) [group accounts” on page 250.](#_bookmark232)

v On a Windows Rational ClearCase LT server, the privileged user is any member of the administrators group on the Rational ClearCase server host.

### The Rational ClearCase server process user

On Windows Rational ClearCase hosts, the **albd\_server** program runs with the identity of a special user account, created during Rational ClearCase site preparation, known as the Rational ClearCase server process user. This user is a member of the Rational ClearCase administrators group and therefore has privileged user status. For more information about the Rational ClearCase server process user, see [“Domain user and group accounts” on page 250.](#_bookmark232) (There is no Rational ClearCase server process user on Rational ClearCase LT.)

### Restricted view access privileges for remote root

Although many Rational ClearCase operations on hosts running Linux or the UNIX system 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.

v When a user logged on as **root** attempts to access a view on a remote host running Linux or the UNIX system, the user’s identity is interpreted as **nobody.nobody** (unidentified user, unidentified group). This interpretation is typical of NFS implementations on Linux and the UNIX system, and it provides a level of security comparable to that provided by NFS. Because Rational 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**.

v Operations that change the user or group ownership of a view-private file, such as the **chown** command on Linux and the UNIX system, fail when run by remote **root**.

## User processes

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

v **User.** The name of the user who starts the process.

v **Primary group.** The primary group of the user who starts the process.

v **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.

## Rational ClearCase objects

The following Rational ClearCase objects are subject to access control:

v VOBs

v Elements and versions

v Types and instances of types, such as labels, branches, and attributes

v Unified Change Management objects, such as projects, activities, and streams

v VOB storage pools

v Views

v 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:

v **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.

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

v **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:

v Read permission, or permission to view the object’s data.

v 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.

v 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 Rational ClearCase object is summarized in [Table](#_bookmark60) 2.

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 are displayed in [Table 2.](#_bookmark60)

|  |  |  |  |
| --- | --- | --- | --- |
| *Table 2. Protection mode for a Rational 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, integers are sometimes used by Rational ClearCase 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 3.](#_bookmark60)

|  |  |  |  |
| --- | --- | --- | --- |
| *Table 3. Protection mode digits for a Rational ClearCase object* | | | |
| **Digit** | **Read permission?** | **Write permission?** | **Execute permission?** |
| 0 | No | No | No |
| 1 | No | No | Yes |
| 2 | No | Yes | No |
| 3 | No | Yes | Yes |
| 4 | Yes | No | No |
| 5 | Yes | No | Yes |
| 6 | Yes | Yes | No |
| 7 | Yes | Yes | Yes |

A sequence of three digits 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:

v The user and groups of the process

v The owner and group of the object

v 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?

v Yes: grant or deny access according to the object’s protection mode for the

**Owner** category.

v No: go to Step [2 on page 59.](#_bookmark61)

1. Does the process have the group ID of the group of the object?

v Yes: grant or deny access according to the object’s protection mode for the

**Group** category.

v No: go to Step [3 on page 59.](#_bookmark61)

1. Grant or deny access according to the object’s protection mode for the **Other**

category.

If an object has no protection mode, access is granted or denied using rules that depend on the type of the object. See the descriptions in [“Access control for VOBs](#_bookmark61) [and VOB objects” on page 59](#_bookmark61) and [“Access control for views and view objects” on](#_bookmark66) [page 64.](#_bookmark66)

In certain cases, a process is granted access to an object by Rational ClearCase 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:

v Elements and versions

v Types and instances of types, such as labels, branches, and attributes

v Unified Change Management objects, such as projects, folders, activities, and streams

v VOB storage pools

## Access control for VOBs

These VOB properties are important for access control:

v **Owner.** The initial owner is the user of the process that creates the VOB.

v **Group.** The initial group is the primary group of the process that creates the VOB.

v **Supplemental group list**. The initial supplemental group list is empty for a VOB created on Windows. On Linux and the UNIX system, 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 Rational 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”](#_bookmark62) and [“Access control for](#_bookmark64) [other VOB objects” on page 62.](#_bookmark64)

### 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”](#_bookmark62) and [“Access](#_bookmark64) [control for other VOB objects” on page 62.](#_bookmark64)

### 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”](#_bookmark62) and [“Access control for other VOB objects” on page 62.](#_bookmark64)

## Access control for elements

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

v **Owner.** The initial owner is the user ID of the process that creates the element.

**Group.** On all supported operating systems, the group of an element must be one that is in the VOB’s group list. When an element is created, its initial group is determined differently on hosts running Linux or the UNIX system than it is on Windows hosts.

* On Linux and on the UNIX system, 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 the process that creates the element and the group list of the VOB, so long as there is exactly one such group.

**Note:** On Windows, if the primary group ID of the process that creates the element is not on the VOB’s group list and the process’s group list includes more than one group that is also in the VOB’s group list, the process cannot create an element in the VOB.

v **Protection mode.** The initial protection mode for a file element is determined differently on hosts running Linux or the UNIX system than on Windows hosts.

– On Linux and on the UNIX system, 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** window in Windows Explorer or Rational 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.

A process whose primary group is one of the VOB’s groups can create any additional elements.

**Note:** On Windows, if the primary group ID of the process that creates the element is not on the VOB’s group list, the process can still create an element if exactly one group appears on both the group list of the process

that creates the element and the group list of the VOB. If the process’s group list includes more than one group that is also in the VOB’s group list, the process cannot create an element in the VOB.

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](#_bookmark63) [write elements.”](#_bookmark63)

### 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.”](#_bookmark63)

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 58.](#_bookmark60)

### 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:

v The process has the user identity of the element’s owner.

v Any of the process’s group identities is the same as the element’s group.

v The process has the user identity of the VOB owner.

v The process has the user identity of a privileged user.

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

v The process has the user identity of the user who checked out the element.

v The process has the user identity of the element’s owner.

v Any of the process’s group identities is the same as the element’s group.

v The process has the user identity of the VOB owner.

v 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” on page 61.](#_bookmark63)

### 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 58.](#_bookmark60) In addition, two special cases can restrict permission to execute an element:

v 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.

v On Linux and the UNIX system, 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:

v Metadata types, such as label types, branch types, and attribute types

v Unified Change Management objects, such as projects, activities, and streams

v Storage pools

v Derived objects

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

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

v **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:

v The process has the user identity of the element’s owner.

v Any of the process’s group identities is the same as the element’s group.

v The process has the user identity of the VOB owner.

v 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.”](#_bookmark65) 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. Temporary access control is also provided by Rational ClearCase 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:

v 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**.

v 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 re-create the release later.

## Access control for views and view objects

Views enable user access to VOB data. As with 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 Rational ClearCase data in a dynamic view requires a process to pass a series of tests:

v It must have access to the view.

v It must have access to the containing directory.

v 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 hosts running Linux or the UNIX system, view access requests from a remote **root** user are treated by Rational ClearCase as requests from the user **nobody.nobody**. See [“Restricted view access privileges for remote root” on](#_bookmark58) [page 56](#_bookmark58) for details.

## Access control for dynamic views

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

v **Owner.** The initial owner is the user of the process that creates the view.

v **Group.** The initial group is the primary group of the process that creates the view.

v **Protection mode.** The initial protection mode for a view is determined one way on hosts running Linux or the UNIX system, 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** window in Windows Explorer or Rational 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 Linux and the UNIX system, the initial protection mode depends on the **umask** of the user who creates the view. A **umask** is a setting supported on Linux and the UNIX system, that specifies that some permissions are not granted when the user creates a file. (For details, see the **umask** reference page on Linux or the UNIX system.) When a user creates a view, Rational 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**, write permission for others is removed by Rational ClearCase.

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 **ls** command on Linux or the UNIX system 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 systems running Linux or the UNIX system, you may need to use **ls –g** to view the group.

**Permission to create views**

Any user can create a view.

### Permission to delete views

Only the view owner or 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 61.](#_bookmark63) 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 67.](#_bookmark69)

An algorithm is used by Rational ClearCase 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 58.](#_bookmark60)

### 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 61.](#_bookmark63) 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](#_bookmark69) [files” on page 67.](#_bookmark69)

An algorithm is used by Rational ClearCase 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 58.](#_bookmark60)

### 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 62.](#_bookmark64) 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 68.](#_bookmark70)

An algorithm is used by Rational ClearCase 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](#_bookmark60) [58.](#_bookmark60)

## 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 Linux and on the UNIX system than they are on Windows.

### Initial owner, group, and protection mode on Linux and the UNIX system

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

v **Owner.** The initial owner is the user of the process that creates the file or directory.

v **Group.** The initial group is the primary group of the process that creates the file or directory.

v **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 setting supported on Linux and the UNIX system that specifies that some permissions are not granted when the user creates a file. (For details, see the**umask** reference page on Linux or the UNIX system.) When a user creates a view-private file or directory, Rational ClearCase begins with a set of permissions that depend on how the file or directory is created. The permissions specified by the user’s umask are then removed by Rational ClearCase. For example, if the user’s umask is 002, write permission for others is removed by Rational ClearCase.

You can use the **cleartool describe** command or the Linux and UNIX

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

v **Owner.** The initial owner is the user of the process that creates the file or directory.

v **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.

v **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** window in Rational 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** window 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 65.](#_bookmark67)

If the containing directory is an element version, the process must have write permission for the element. See [“Permission to write elements” on page 61.](#_bookmark63) 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](#_bookmark69) [files.”](#_bookmark69)

### 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 65.](#_bookmark67)

If the containing directory is an element version, the process must have write permission for the element. See [“Permission to write elements” on page 61.](#_bookmark63) 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](#_bookmark69) [files.”](#_bookmark69)

### 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 65.](#_bookmark67)

An algorithm is used by Rational ClearCase 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](#_bookmark60) [to VOB and view data” on page 58.](#_bookmark60)

### 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 65.](#_bookmark67)

An algorithm is used by Rational ClearCase 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 58.](#_bookmark60)

### 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 65.](#_bookmark67)

An algorithm is used by Rational ClearCase 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 58.](#_bookmark60)

## 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 similar to 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 66.](#_bookmark68)

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. An algorithm is used by Rational ClearCase 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](#_bookmark60) [view data” on page 58.](#_bookmark60)

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, the DO is converted to a view-private copy.

## Rational ClearCase and native file system permissions

Data for VOBs and views is maintained by Rational ClearCase in ordinary directories, the VOB storage directory and the view storage directory, in the native file system on each VOB and view host. All access to these directories is managed by Rational ClearCase; 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.

**Attention:** 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 199.](#_bookmark191)

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 Rational 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 Rational ClearCase control.

As with a dynamic view, a snapshot view also has a view storage directory, which may or may not be located within the snapshot view directory. The view storage directory for a snapshot view is created and maintained by Rational ClearCase, just as it is for a dynamic view. Never change native file system permissions on the view storage directory for a snapshot view.

# Chapter 6. VOB administration

VOB administration tasks include the following:

v Configuring VOB hosts

v Creating VOBs

v Enabling access to VOB data

v Importing data into a VOB from another configuration management system

v Backing up and recovering VOB data

v Managing VOB storage

v Monitoring VOB integrity

## Access to VOB data and metadata

A VOB is a repository for:

1.) data (versioned artifacts such as files and directories) and

2.) metadata (branches, labels**, event records**, and so on).

**VOB data** must be accessed through a view. 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.

VOB data can be created by users or imported from another configuration management system.

**VOB metadata** can be accessed directly by using Rational ClearCase GUIs and **cleartool** commands.

VOB metadatacan be created in several ways:

v As an indirect result of a command that creates a VOB or an object in a VOB.

1. For example, commands that create VOBs also create VOB attributes and UUIDs;
2. the **cleartool mkbranch** command creates a branch and local instance of a branch type if needed.

v As the direct result of a command that creates or manipulates a specific type of metadata.

1. for example the **cleartool mk***object***type** commands (for example, **mkeltype** and **mkbrtype**), which create specific type objects

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

## VOB attributes

v A VOB with the **UCM VOB** (sometimes displayed as **sumvob**) attribute holds components that are part of a UCM project.

v 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.

Any single VOB can be an administrative VOB, a PVOB, and either an ordinary or a UCM component VOB Larger communities may require multiple administrative VOBs or PVOBs.

## VOB schema versions

Each VOB has a database schema version that denotes the

1. format of the VOB database and
2. determines the types of data and metadata that the VOB can store.

Rational ClearCase v7.0 supports only schema version 54.

* Schema version 54 is supported on all supported platforms
* provides support for VOB database files larger than 2 GB.

### Upgrading your VOB schema version

Rational ClearCase v7.0 supports only schema version 54. If you have an existing VOB that is at schema version 53 or lower, you need to upgrade the VOB to schema version 54. To upgrade your VOB to schema version 54:

1. Install Rational ClearCase v7.0 on the VOB server host.
2. After the Rational ClearCase installation is complete, all VOBs on the host that are at schema 53 will be inaccessible until you reformat them with the **cleartool reformatvob** command.

## VOB feature levels

* A feature level is an integer that defines the set of Rational ClearCase features that a VOB supports.
* Whenever a Rational 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 Rational ClearCase release upgrades when VOBs are replicated (with Rational ClearCase MultiSite) to server hosts that run different Rational ClearCase releases.

### To display the VOB feature level

To display the feature level of a replica family,

**cleartool describe replica:***replica-name***@***vob-tag*

**cleartool describe replica:tokyo@\dev**...

feature level: 5

...

To display the feature level of a VOB family,

**cleartool describe vob:***vob-tag*.

**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: 5

...

**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.
3. The command lists each VOB served by the host.
4. 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 Rational ClearCase installed on the host.

When you use Rational ClearCase 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.

## VOB server host configuration guidelines

A host on which a **vob\_server** process runs is a VOB host.

A VOB host runs a number of server processes for each VOB it supports.

## Physical memory

The memory requirements for a VOB server need to be sufficient for:

v The data and instruction space needs of Rational ClearCase processes (1 albd\_server, m vob\_servers and 5m vobrpc\_servers - where m is the number of VOBs, and n db\_servers where n is the maximum number of simultaneous users at peak).

v Sufficient file system cache to accommodate the working set of VOB database pages accessed during normal daily Rational ClearCase activity.

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: View\_server** processes should not be run on a VOB host (see [“Minimize](#_bookmark212) [process overhead” on page 225).](#_bookmark212)

## Swap space

On a VOB server host, the recommended amount of swap space is equal to:

v The maximum number of db\_servers multiplied by the size of each db\_server process (approximately 5 MB), plus ndb\_servers\*5

v The maximum number of vobrpc\_servers multiplied by the size of each vobrpc\_server process (approximately 10 MB) no.of vobrpc\_servers\*10

Where the maximum number of db\_servers is equal to the maximum number of **simultaneous** users of all the VOBs on the VOB server, and the maximum number of vobrpc\_servers is equal to the number of VOBs hosted on the VOB server multiplied by 5.

For example, say that a Solaris VOB host needs to accommodate 200 VOBs and 300 simultaneous users. The recommended amount of swap space equals 200\*10MB = 2 GB, plus 300\*5 MB = 1.5 GB, or 3.5 GB.

## Disk capacity

A VOB database (and, on Windows, the entire VOB storage directory) must fit in a single disk partition. Although there is no general guidance for estimating how large a VOB will be, [Appendix D, “Estimating VOB size,” on page 271](#_bookmark251) provides some guidelines that may be useful is establishing VOB server disk capacity requirements.

Use a **high-performance disk subsystem 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.

**striping** helps overall performance and more than makes up for any degradation caused by mirroring.

## Processor capacity

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 Rational 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.

## Adjusting kernel resources on Linux and the UNIX system

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

v **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.

v **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.

## Creating VOBs

## 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.

v On hosts running Linux or the UNIX system, the partition where the VOB storage location is created must be exported so that clients running Linux or the UNIX system 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.

v 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 Rational ClearCase administrators group.

## Creating server storage locations for VOBs

Server storage locations allow administrators to designate specific hosts and disk volumes as preferred locations for VOB storage.

You can specify one or more VOB storage locations for each region you have defined.

Server storage locations for VOBs should be created on a disk partition that has

1. plenty of room for VOB database growth and is
2. accessible to all hosts in the region

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

v On a computer running any supported operating system, you can use the

**cleartool mkstgloc** command to create server storage locations.

v server setup wizard creates server storage locations.

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

1. Start the Rational ClearCase Administration Console.
2. Navigate to the **Storage Locations** subnode of the **ClearCase Registry** node and do any of the following:

v To create a new storage location, click **Action > New > Server Storage Location**.

v To change the properties of an existing storage location, select one and click

###### Action > Properties.

v To remove an existing storage location, select one from the list and click

###### Action > Remove Server Storage Location.

**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 region on Linux or the UNIX system.

**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.

If storage location that resides on a NAS device run on the host specified in the **–host** option to **mkstgloc**.

**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

command-line interface or GUI tools to create a VOB.

1. **Choose a location for the VOB storage directory.**

You can use an existing VOB server storage location or create a new server storage location, or use any other directory that has the proper characteristics for VOB storage.

1. **Log on to the VOB host**.

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.

1. **Choose a VOB tag**
2. **Create the VOB.**

When you run the following command on Windows **host pluto**, the command 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

1. **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 .**

### To create a VOB on a NAS device

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.**

The VOB is served by a **vob\_server** process running on Rational ClearCase host **vobsvr1** (a computer running Linux or the UNIX system) 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**

## 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.**

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.

**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.

v On computers running Linux or the UNIX system, all public VOBs are mounted as a group when Rational ClearCase starts. To defeat this behavior, use the **noauto** mount option at VOB creation time (see **mkvob –options**).

v 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

**Note:** On all supported operating systems on which 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 You must also supply the Rational ClearCase registry password.

## 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 Rational ClearCase data model. It is useful to establish an administrative VOB hierarchy into which new VOBs can be placed in order to simplify type administration.

**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.

1. GUI: With the VOB Creation Wizard, you can specify the administrative VOB for each new VOB that you create.
2. Cmd: 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.**

Changing server host

The following procedure replaces the VOB host running Linux or the UNIX system for the VOB /vobs/libpub with a different Rational 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.

###### Lock the VOB.

1. **Unmount the VOB** as needed on dynamic view hosts.
2. (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

1. **Remove the VOB tag and unregister the VOB.**

###### cleartool rmtag –vob –all /vobs/libpub

**cleartool unregister –vob /net/nasdevice/vobstg/libpub.vbs**

1. **Terminate the VOB’s server processes on the current host**.

v On a Windows host, stop and restart Rational ClearCase.

1. **Register the VOB and create a new VOB tag**.

###### cleartool register –vob –host vobsvr1

**–hpath /net/nasdevice/vobstg/libpub.vbs \**

**–gpath /net/nasdevice/vobstg/libpub.vbs /net/nasdevice/vobstg/libpub.vb**

**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**

1. **Unlock the VOB.**
2. **Mount the VOB** as needed on dynamic view hosts.
3. (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

## 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.

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:

v It is the same type of host.

v It has network connectivity with the VOB host.

v The VOB storage directory meets the criteria defined in [“Access requirements for](#_bookmark76) [VOB storage”](#_bookmark76)

## The automounter on Linux or the UNIX system does not use the hosts map

Use –host, -gpath, -hpath for vobs, views

## 

## 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.

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:

v 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.

v 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

v 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 Linux and the UNIX system

By default, the viewroot and VOB file systems are mounted with setuid and setgid disabled.

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 installations, and you must stop and restart Rational ClearCase on the host after running the script. For more information, see the discussion of **site\_prep** in the *IBM Rational ClearCase, ClearCase MultiSite, and ClearCase LT 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.

## To remove a VOB tag

Removing a VOB’s tag makes the VOB inaccessible. Use the Rational ClearCase Administration Console or the **cleartool** Rational 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 that it contains is no longer valuable.

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 Rational ClearCase Administration Console.
3. Stop and restart Rational 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.

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 Rational 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 (Rational 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 Rational ClearCase MultiSite have different replica UUIDs.

**.identity** On hosts running Linux or the UNIX system, a subdirectory whose files establish the VOB’s owner and group memberships. See [“Access control for](#_bookmark61) [VOBs and VOB objects” on page 59.](#_bookmark61)

**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 Rational ClearCase 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 systems running Linux or the UNIX system, 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 Rational 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 Rational 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](#_bookmark90) [storage for a VOB” on page 90.](#_bookmark90)

**Note:** In Rational 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, the cleartext pool is checked first by Rational ClearCase, and a new version is only constructed 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](#_bookmark171) [page 176.](#_bookmark171)

### 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.

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

**Note:** The DO pools of Rational ClearCase LT VOBs are always empty because Rational 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:

v Version-control information: elements, their branch structures, and their versions

v Metadata associated with the file system objects: version labels, attributes, and so on

v Event records and configuration records, which document changes in artifacts and related development activities

v Type objects, which are involved in the implementation of both the version-control structures and the metadata

v (PVOBs only) UCM objects: folders, projects, streams, activities, components, baselines, and so on

The permanent contents of artifacts that are under Rational 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 (Linux and the UNIX system)**

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 Rational 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 previous 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. The **reformatvob** command preserves the previous database by renaming it. Thus, a VOB storage directory may contain earlier (and usually unneeded) VOB database subdirectories, with names such as db.0318. If **reformatvob** is interrupted, it may leave a partially reformatted database with the name db.reformat.

## The .identity directory

On platforms running Linux or the UNIX system, 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

1. the number of developers who use the VOB and
2. the rate at which they create and change artifacts in it.

VOB storage management includes three principal tasks:

v Monitoring the storage each VOB consumes

v 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.

v Creating additional storage capacity by:

* adding new storage pools to a VOB
* relocating data from one VOB to another
* moving an entire VOB to another partition or host.

## Using the scheduler

The Rational ClearCase scheduler runs several jobs:

v Daily data gathering on disk space used by VOBs

v Weekly data gathering on disk space used by shared derived objects

v Daily scrubbing of VOB storage pools by the **scrubber** utility

v Weekly scrubbing of VOB databases by the **vob\_scrubber** utility

v 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.](#_bookmark164)

## 

## Monitoring VOB storage

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

v In the Rational ClearCase Administration Console, the VOB storage directory node for each **VOB** subnode of a Rational 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.

v The **cleartool space –vob** command shows current and historical disk space use for a VOB.

v 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.](#_bookmark164)

## Scrubbing

Two utility programs, **scrubber** and **vob\_scrubber**, are included in Rational ClearCase.

You can use these utilities to control the growth of the VOB storage pools and database.

The Rational 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.

## 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:

v Even if an element is no longer needed for ongoing work, you may need it to reproduce and maintain earlier work.

v **rmelem** removes the element’s name from all directory versions in which it was ever cataloged. This erasing of history means that the element is not displayed in listings or comparisons of previous 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.

v 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 161.)](#_bookmark157)

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:

v Versions that are unrelated to branching (not located at a branch point and not the first or last version on a branch)

v Versions that have no metadata annotations, such as version labels, attributes, or hyperlinks

## 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:

v Create new elements, and then cancel the checkout of directory in which they were created

v Delete the last reference to an element by using the **rmname** command

v 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:

v It cannot be checked out.

v Its contents can be modified even though it is not checked out.

v No branches can be created within it.

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

v If you need an element in lost+found, move it to another directory by using

###### cleartool mv.

v You can use **cleartool rmelem** to remove elements from lost+found if you are sure that 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 Rational 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:

v Creating additional storage pools in a VOB storage directory

v Using symbolic links to create additional pools in other directories or on other hosts

v 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

## Creating remote storage pools on hosts running Linux or the UNIX system

On hosts running Linux or the UNIX system, which support symbolic links to file systems on other hosts running Linux or the UNIX system or to 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 Rational 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:

v 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 149.](#_bookmark145)

v VOB tags for VOBs with remote storage pools must include additional information if they are to be accessed by Windows computers. See [“Windows](#_bookmark220) [tags for VOBs on Linux or the UNIX system with remote storage pools” on page](#_bookmark220) [235.](#_bookmark220)

v 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.

v 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:

v **Source pools.** These pools store the most critical data: checked-in versions of file elements. Traffic to and from these pools is relatively light, but data integrity is very important. Source storage pools should be kept 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 re-create the lost versions. If you decide to use remote source pools, choose a robust file server for which you provide frequent, reliable data backups.

v **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.

v **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 earlier derived objects can sometimes be difficult. The DO pools of Rational ClearCase LT VOBs are always empty, because Rational 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 hosts running Linux or the UNIX system, 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 cleartext pools 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 Linux or on the UNIX system, 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".

1. **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".

1. **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 host hosts running Linux or the UNIX system:

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"

###### Lock the VOB.

1. **Copy the contents of the storage pool**. The storage pool is a standard directory on Linux or the UNIX system. 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**

1. **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.

###### Unlock the VOB.

1. **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 VOBs on](#_bookmark220) [Linux or the UNIX system with remote storage pools” on page 235.](#_bookmark220)

# Chapter 7. VOB datatypes and administrative VOB hierarchies

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 such as 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 Rational ClearCase commands. Others must be created explicitly by users or (more often) administrators. You must use commands such as **cleartool describe** and various Rational 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:

v 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.

v 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 systems

running Linux or the UNIX system, this string is

© Copyright IBM Corp. 1992, 2005 **93**

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:

v The name of DO and the name of its containing directory

v The DO’s unique identifier

v Shopping information for the DO (used by **clearmake** and **omake**)

DOs are not created on Rational ClearCase LT because they can 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 in Rational ClearCase LT because CRs can 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 such as **describe**. For example, to describe a branch type named **v4\_patch**, use the **brtype** mnemonic as shown.

###### 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\_7.0** to a particular version does not make a copy of the **BASELEVEL\_7.0** 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\_7.0** to **BL7.0** 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](#_bookmark96) [types” on page 96.](#_bookmark96)

**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:

v **mkattype** creates or modifies an attribute type object

v **mkbrtype** creates or modifies a branch type object

v **mkeltype** creates or modifies an element type object v **mkhltype** creates or modifies a hyperlink type object v **mklbtype** creates or modifies a label type object

v **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 Rational 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:

v It contains at least one global type.

v 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,](#_bookmark97) 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.

**\admin**

**\admin\_re \admin\_dev**

**\re \dev**

= AdminVOB hyperlinks

= global type RELEASE-2.1

= local copy RELEASE-2.1

= global type RELEASE-3.0

= local copy RELEASE-3.0

= global type PATCH-2.1

= local copy PATCH-2.1

*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"

1. 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".

1. Link the VOB you removed from the hierarchy in Step [1 on page 98](#_bookmark98) 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 Rational 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.

v VOB **\dev**, whose administrative VOB is \**admin\_dev**

**cleartool describe vob:\dev**

versioned object base "\dev"

...

Hyperlinks:

AdminVOB -> vob:\admin\_dev

v 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

v 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:

v 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.

v 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.

v 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”](#_bookmark161) [on page 165.](#_bookmark161)

v 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](#_bookmark99) [administrative VOB becomes unavailable.”](#_bookmark99)

## If an administrative VOB becomes unavailable

When an administrative VOB (or PVOB) is unavailable, all other VOBs in its administrative hierarchy are affected. Many operations on global types and their local copies require access to the administrative VOB and will fail if that access cannot be obtained. For example:

v Attempts to lock a VOB that has an **AdminVOB** hyperlink to an unavailable VOB fail.

v Attempts at other VOBs in the hierarchy to create instances of a global type defined in an unavailable VOB produce the following error:

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

*VOB-tag*

v The output of **cleartool describe** for a VOB that has an **AdminVOB** hyperlink to an unavailable VOB may not show the hyperlink.

**Note:** An administrative VOB does not have to be mounted to be available for these operations, but it must be registered and must have a tag in the region from which the operation is attempted.

An administrative VOB (like any VOB) can become unavailable for a number of reasons, including the following:

v The VOB server host is down or disconnected from the network.

v Rational ClearCase has been stopped on the VOB server host.

v The VOB is replicated but has no replica in the current registry (see [“Replicated](#_bookmark106) [administrative VOB hierarchies” on page](#_bookmark106) 106).

v The VOB is being reformatted. When reformatting one or more members of an administrative VOB hierarchy, start at the bottom of the hierarchy and work your way up so that no member of the hierarchy is accessible while its administrative VOB is not.

## 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 Rational 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, contact IBM support for Rational products.

### 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

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

**cleartool rmhlink AdminVOB@40@\dev**

Removed hyperlink "AdminVOB@40@\dev".

**Note:** A warning message will be produced if you attempt to remove a hyperlink from a UCM PVOB to an AdminVOB. The hyperlink will be removed, but the warning message includes what command you can use to replace the hyperlink, if needed. Following is an example of this warning message:

**cleartool rmhlink AdminVOB@40@\dev**

cleartool: Warning: An AdminVOB hyperlink to a UCM PVOB is being removed. This can cause serious problems with UCM. If desired, this hyperlink may be replaced using the command:

cleartool mkhlink AdminVOB vob:\dev vob:\vobs\pvob

### 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**

...

1. 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

1. 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 Rational ClearCase Administration Console, related hyperlinks are cleaned up automatically:

v All **AdminVOB** hyperlinks to the VOB are removed.

v All **GlobalDefinition** hyperlinks connecting to global types in the VOB are removed.

v 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 Rational 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](#_bookmark96) [objects” on page 96)](#_bookmark96) 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:

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

v A checkout operation that creates a branch (because of an auto-make-branch rule) creates a local copy of the global branch type.

v 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), a local copy of the global type is not created by Rational ClearCase, 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, the global type in the administrative VOB hierarchy is searched by Rational ClearCase.

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: v 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.)

v 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

1. 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".

1. 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](#_bookmark181) [hyperlinks” on page 187.](#_bookmark181)

## 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](#_bookmark107) 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.

###### Boston

**San Francisco**

**\re**

**BOS@\admin**

**SFO@\admin**

**\admin\_re**

**BOS@\admin\_dev**

**SFO@\admin\_dev**

**BOS@\dev**

**SFO@\dev**

replication

*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, all members of the hierarchy should be synchronized at the same time. If you do not, users may have trouble accessing type objects.

**Note:** In [Figure 5,](#_bookmark107) 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](#_bookmark107) 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

1. 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

1. 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".

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

**multitool syncreplica –export –fship SFO@\dev**

1. 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

*IBM Rational ClearCase MultiSite Administrator’s Guide*.

# Chapter 8. Moving VOBs and relocating VOB data

Rational ClearCase includes tools for moving data and metadata from one VOB to another and for moving entire VOBs from one disk or host to another.

## Relocating elements to another VOB

The **cleartool relocate** command moves elements from one VOB to another one that has a tag in the same registry region. You can relocate file or directory elements. When you relocate a directory element, you relocate all of the elements the directory contains.

A **relocate** operation is appropriate when:

* you need to reorganize VOB contents to reflect changes in system architecture or organizational structure
* 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.

**Moving VOBs**

Special procedures that maintain:

1.)the integrity of VOB data and

2.) preserve permissions and ownership on VOB storage directories.

v Moving a VOB to another disk partition on Windows or to another Windows host in the same domain.

v Moving a VOB to a Windows host in a different domain.

v Moving a VOB to another disk partition on Linux or the UNIX system, or to another host running Linux or the UNIX system with the same architecture (binary data format).

v Moving a VOB to a host running Linux or the UNIX system, with a different architecture.

v Moving a VOB from a Windows host to a host running Linux or the UNIX system.

v Moving a VOB from a host running Linux or the UNIX system to a Windows host.

**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

1.) 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.

The VOB cannot be used in its new location until you fix the protections on the VOB storage directory.

2.) 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.

3.) Scrub the VOB’s pools to reduce their size before moving the VOB.

4.) If you are moving a UCM-enabled VOB, you will need to move your PVOB before you move your component VOB.

5.) 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.

## 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 all replicas should 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.)

1. **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".

**Moving a VOB on Windows**

This section describes two common procedures:

v 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.

v 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**.

###### Lock the VOB.

1. **Stop Rational ClearCase** on the VOB server host.
2. **Copy the VOB storage directory, preserving all ownership information**.

Use the Rational ClearCase **ccopy** utility, located in*ccase–home–dir*\etc\utils\ccopy.exe for this purpose. You must run **ccopy** on the host to which the VOB is being moved.

C:\ClearCaseStorage\VOBs> **net use E: \\vobsvr01\vobstg**

C:\ClearCaseStorage\VOBs> **ccopy libpub.vbs E:\libpub.vbs**

1. **Restart Rational ClearCase** if you have copied the VOB storage directory to a new location on the same VOB server host.
2. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the Rational ClearCaseAdministration 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**

###### Unlock the VOB.

1. **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 incorrect and must be changed (mapped) to SIDs that are valid in the new domain. Rational ClearCase includes a utility program, **vob\_sidwalk**, that provides a flexible means of mapping SIDs after you move a VOB to a different domain. 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 run 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 Rational 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”](#_bookmark72) [on page 72.](#_bookmark72)

###### Log on to the VOB server host as the VOB owner or privileged user.

1. **Lock the VOB.** This ensures that no new VOB objects are created while you complete Step [4 on page 117.](#_bookmark116)
2. **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

Create 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 will need it in Step [10 on page 1](#_bookmark116)17.)

1. **Stop Rational ClearCase** on the VOB server host.
2. **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 /E**

**Note:** Because the existing VOB storage directory ACLs are not valid in the new domain, you may use **xcopy** or another copy utility that does not preserve ACLs for this step.

1. **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

1. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the Rational 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 the old registration and tag for \libpub should be removed, because this data will not be valid after the move.

1. **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.
2. **Create a map file.** Open the SID file generated in Step [4 on page 117](#_bookmark116) (\\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 looks 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](#_bookmark244) [owner” on page 262.](#_bookmark244)

1. **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

1. **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.
2. **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 on page 117:](#_bookmark116)

*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.

1. **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

1. **Verify that all clients in the new domain can access the VOB.** Unlock the VOB if it is still locked.
2. **Verify that all Rational 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 run by a more privileged user (VOB owner, member of the Rational ClearCase administrators group).

## Moving a VOB on Linux or the UNIX system

This section describes two common procedures for moving VOBs on Linux or the UNIX system:

v Moving a VOB on a VOB server host running Linux or the UNIX system that has the same architecture (binary data format). You can also use this procedure to move a VOB to another partition on a host running Linux or the UNIX system.

v Moving a VOB to a host running Linux or the UNIX system that has a different architecture.

For clarity, the procedures in this section use an example:

v The current location of the VOB storage directory to be moved is

/vobstore/libpub.vbs, on a host named **sol**.

v The VOB tag is /vobs/libpub.

v 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 are accessible after the move.

###### 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**.

1. **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 new VOB host can use the same global pathnames used by the current VOB host to access each remote storage pool:

v If you are moving the VOB to another location on the same host, these global pathnames will continue to be valid.

v 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 Linux or from the UNIX system to Windows (see [“Moving a VOB from Linux or the UNIX system to](#_bookmark123) [Windows” on page 125),](#_bookmark123) 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](#_bookmark125) [network-attached storage” on page 128).](#_bookmark125)

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 **find** command on Linux or the UNIX system 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

1. **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 commands in this step, which are supported on Linux and the UNIX system, 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–)**

1. **Verify that the VOB has no remote pools.** Stop and restart Rational 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.

1. **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 **Registry Regions** node of the Rational ClearCase Administration Console. The **Properties** page for the VOB tag has a **Mount Options** page that allows you to edit the split pool map if you are logged in as a member of the Rational ClearCase administrators group. If you cannot use the Rational ClearCase Administration Console, use **cleartool rmtag** and **mktag** to remove the VOB tag and re-create it without a split pool map.
2. **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.

## Moving a VOB between hosts running Linux or the UNIX system that have the same architecture

You can use the following procedure to move a VOB to another disk partition on a host running Linux or the UNIX system, or to another host running Linux or the UNIX system that has the same architecture.

1. **Log on to the VOB server host.** Log on as the VOB owner or privileged user.

###### Lock the VOB.

1. **Stop Rational ClearCase** on the VOB server host.
2. **Copy the VOB storage directory**. The procedure you use depends on whether you are 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 platforms running Linux or the UNIX system. 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 on Linux or the UNIX system, use the **mv** command.

1. **Restart Rational ClearCase** if you have copied the VOB storage directory to a new location on the same VOB server host.
2. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the Rational 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**

1. Unlock the VOB
2. Verify that all clients can access the VOB at the new location.

## Moving a VOB between hosts running Linux or the UNIX system that have different architectures

Use the following procedure to move a VOB to a host running Linux or the UNIX system that has a different architecture. The procedure is similar to that described in [“Moving a VOB between hosts running Linux or the UNIX system that have the](#_bookmark119) [same architecture” on page 121,](#_bookmark119) 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.

* 1. **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 platforms running Linux or the UNIX system. Also, the **rsh** command may have a different name, such as **remsh**, on some platforms. See the reference pages for your operating system.

* 1. **Terminate the old VOB server processes**. You can either stop and restart Rational 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 **kill** command to terminate any such processes on Linux or the UNIX system.
  2. **Log on to the new VOB server host.** Log on as the VOB owner or privileged user.

###### Re-create the VOB database from the dump files:

**cleartool reformatvob –load /src/vobstore/libpub.vbs**

* 1. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the Rational 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**

* 1. Unlock the VOB.
  2. Verify that all clients can access the VOB at the new location.

## Moving a VOB to a different operating system type

When you move a VOB from a Windows host to a host running Linux or the UNIX system 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:

v Run **vob\_sidwalk** before the move to capture information about ownership of VOB objects.

v Use **reformatvob** to dump the VOB database into a portable form.

v Copy the VOB storage directory, which includes the dumped database, to the new host.

v Use **reformatvob** to load the VOB database in the proper binary format.

v Reset the file system protections on the VOB storage directory after the move.

v Remap the SIDs (or, on Linux or the UNIX system, 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.

## Moving a VOB from Windows to Linux or the UNIX system

For clarity, the procedures in this section use an example:

v 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.

v The new location for the VOB storage directory is /vobstg/libpub.vbs on the host running Linux or the UNIX system, **vobsvr2**. VOB tag /vobs/libpub will be created for this VOB.

To move a VOB from Windows to Linux or the UNIX system:

###### Log on to the Windows VOB server host as the VOB owner or privileged user.

1. **Lock the VOB.** This ensures that no new VOB objects are created while you complete Step [3.](#_bookmark121)
2. **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

Create 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 on page 125.](#_bookmark123)

1. **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.

1. **Copy the VOB storage directory**. Use any file system copy utility to copy the entire VOB storage directory to the host running Linux or the UNIX system. This example assumes that the target 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 hosts running Linux or the UNIX system, you may use **xcopy** or another copy utility that does not preserve ACLs for this step.

1. **Terminate the VOB’s server processes on Windows**. Stop and restart Rational ClearCase on the Windows VOB server host (**\\vobsvr-nt** in our example).

###### Log on to the VOB server host running Linux or the UNIX system.

1. **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**

1. **re-create the VOB database from the dump files**:

# **cleartool reformatvob –load /vobstg/libpub.vbs**

1. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the Rational 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**

1. **Create a map file.** Open the SID file generated in Step [3 on page 123](#_bookmark121) (/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 new 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:3.01050*37...* | akp |  |  |  |

Although this example shows a user name that is the same on Linux or the UNIX system 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 Linux or the UNIX system.

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](#_bookmark244) [owner” on page 262.](#_bookmark244)

1. **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**

1. **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*).
2. **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 Step1[1 on page 124](#_bookmark122) :

*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.

###### 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

1. **Verify that all clients can access the VOB at the new location.** Unlock the VOB if it is still locked.
2. **Verify that all Rational 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 Linux or the UNIX system to Windows

For clarity, the procedures in this section use an example:

v The current location of the VOB storage directory to be moved, on the host running Linux or the UNIX system, is /vobstg/libpub.vbs ,**vobsvr2**. The VOB tag for this VOB is /vobs/libpub.

v The new location of the VOB storage directory is C:\ClearCaseStorage\VOBs\libpub.vbs on the Windows host **vobsvr-nt**. The VOB tag **\libpub** will be created for this VOB.

To move a VOB from Linux or the UNIX system to Windows:

**Note:** If the VOB has remote storage pools, you must first consolidate those pools under the VOB root directory. Symbolic links that are supported on Linux and the UNIX system 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 120](#_bookmark118) 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.](#_bookmark123)
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

Create 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 on page 127.](#_bookmark124)

1. **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.

1. **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 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 hosts running Linux or the UNIX system, you can use **xcopy** or for this step.

1. **Terminate the VOB’s server processes on Linux or the UNIX system**. You can either stop and restart Rational 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**kill** command to terminate any such processes on Linux and the UNIX system.
2. **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

1. **Re-create the VOB database from the dump files**. Use the cleartool

**reformatvob** command:

**cleartool reformatvob –load C:\ClearCaseStorage\VOBs\libpub.vbs**

1. **Replace the VOB object and tag** with new ones that reference the new VOB storage directory. Use the Rational 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**

1. **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.
2. **Create a map file.** Open the SID file generated in Step [4 on page 117](#_bookmark116) 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 Linux or the UNIX system, this procedure can also be used to map a user or group name on Linux or the UNIX system 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](#_bookmark244) [owner” on page 262.](#_bookmark244)

1. **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**

1. **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.
2. **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 on page 117:](#_bookmark116)

*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.

1. **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

1. **Verify that all clients in the region can access the VOB.** Unlock the VOB if it is still locked.
2. **Verify that all Rational 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 Linux or the UNIX system, 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 run by a more privileged user (VOB owner, member of the Rational 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 Rational ClearCase host running Linux or the UNIX system to a NAS device, follow the procedures in [“Moving a](#_bookmark117) [VOB on Linux or the UNIX system” on page 119,](#_bookmark117) 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 host **ccvobsvr1**, which is running Linux or the UNIX system.

###### 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](#_bookmark115) [Windows” on page 116,](#_bookmark115) 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 computer running Linux or the UNIX system can have one or more of its storage pools on a remote host running Linux or the UNIX system, accessed through a 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. You do not need to consolidate the remote pools as long as they are accessible to the VOB server host, but doing so simplifies VOB backups and other administrative tasks. See [“Consolidating remote pools” on](#_bookmark118) [page 120.](#_bookmark118)

# Chapter 9. View administration

Introduction to views and view administration

development environment requires one or more views. A view provides a workspace where users access versions of file and directory elements that are under Rational ClearCase control. Views can also contain

view-private file system objects (such as ordinary files and directories) that are not under Rational ClearCase control.

v View creation and access control

v Backing up and recovering views

v Moving, removing, and managing the storage used by views

There are three kinds of views:

v Dynamic

v Snapshot views

v Web views are similar to snapshot views, but are created and accessed through the Rational ClearCase Web interface.

## Dynamic views

**Note:** Dynamic views are not supported by Rational ClearCase LT.

A dynamic view is an MVFS directory that enables dynamic access to VOB elements.

v Selected versions of elements (actually stored in VOB storage pools)

v Files that are being modified (checked-out file elements, stored in the view’s private storage area)

v Directories that are being modified (checked-out directory elements, maintained in the VOB database)

v **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

**v Shared derived objects built in another view** and winked in to this view (stored in VOB storage pools)

v View-private objects: miscellaneous files, directories, and links that are not under Rational ClearCase control and are displayed only in this view (stored in the view’s private storage area).

## The view root

Every dynamic view on a host is displayed as a subdirectory of the host’s view root directory. On computers running Linux or the UNIX system, 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:

v On Linux or the UNIX system, edit the file *ccase–home–dir*/etc/clearcase to change this line, which defines the viewroot on Linux or the UNIX system. VIEWPATH=”/view”

v On Windows, use the ClearCase program in Control Panel. The view root is specified as **View network name** on the **MVFS** page.

## 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 Rational 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 Linux or on the UNIX system, 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 platforms running Linux or the UNIX system, .s can be either a local directory or a symbolic link to a directory on another computer running Linux or the UNIX system or a network-attached storage device.

The private storage area holds several kinds of objects:

v **View-Private objects.** A view-private object is an ordinary file system object, such as a file, directory, or link, contained in a view. View-private objects exist only within the view’s private storage area and are not stored in a VOB.

v **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 similar to a view-private file, except that there is a corresponding object in the VOB: a placeholder version with the special CHECKEDOUT version label.

v **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 use 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.

v **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.

v **Configuration records.** The file view\_db.crs\_file in the .s subdirectory stores the configuration records of derived objects built in the view.

v **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 Rational 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 non-shareable

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 Rational ClearCase LT. Snapshot and Web views do not use the MVFS.The multiversion file system (MVFS) is a feature of Rational 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 computers running Linux or the UNIX system, 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 Rational 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:

v Files

v Directories

v Symbolic links (on Linux or the UNIX system only)

You cannot create other file types, such as special files supported on Linux or the UNIX system, within a dynamic view.

## The MVFS and case-sensitivity

On Windows, the MVFS can be configured to perform name lookups that are or are not case-sensitive. In the default configuration, the MVFS performs name lookups that are not case-sensitive, which is the behavior that Windows applications expect. When using dynamic views in mixed networks of computers running different supported operating systems, you may need to change

case-sensitivity setting of the MVFS on Windows. See [“Case-sensitivity” on page 49](#_bookmark52) for details.

## 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.

## Snapshot views

Any snapshot view storage directory contains two subdirectories:

v **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.

v **Administrative directory.** The admin directory contains data on disk space used by the view. Periodically, the Rational 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.

## 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 Rational 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 Linux or the UNIX system.)

view.stg or a generated directory name

A directory used to maintain the view. (.view.stg on Linux or the UNIX system.) 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 Rational 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 Linux or the UNIX system, 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 Rational 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 client programs locate snapshot views

Information that Rational ClearCase client programs use to locate snapshot view directories is stored in per-user snapshot view registries. On Windows, this information is stored in the Windows registry of the host from which the user created the snapshot view. On Linux or the UNIX system, this information is stored in a file named .ccase\_svreg in the user’s home directory.

**Note:** If .ccase\_svreg becomes corrupted or has an access mode that does not allow reading or writing, the user’s snapshot views may become inaccessible. For information on how to regenerate this file, see *Working in Base ClearCase*.

## 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

## Remote view storage

As with 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:

v 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).

v You can locate some or all of a view’s storage on a certified network-attached storage device.

v On a host running Linux or the UNIX system, a view’s private storage area (but not its database) can be located remotely and accessed through a symbolic link. This arrangement resembles the remote VOB storage pool facility, discussed in

[“Creating remote storage pools on hosts running Linux or the UNIX system” on](#_bookmark90) [page 90,](#_bookmark90) but the facility for views is less elaborate.

**Note:** A Windows computer cannot use a view hosted on Linux or the UNIX system 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, first create a server storage location for this purpose (see [“Creating server storage locations on a NAS device” on page](#_bookmark77) [77).](#_bookmark77) 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 Rational ClearCase host **ccviewsvr-ux** (a computer running Linux or the UNIX system) 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.

## Determining whether a dynamic view on Linux or the UNIX system has remote private storage

The procedures in this section do not move any remote pools that may be associated with views on hosts running Linux or the UNIX system. Before you move a dynamic view on a host running Linux or the UNIX system, determine whether the view has symbolically linked private storage. (Snapshot views do not support symbolically linked view-private storage.) Use the **ls** command to list the view’s .s directory on Linux or the UNIX system. 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

1. **Stop the view’s server process**. Use the **cleartool endview** *view\_tag* command with the **–server** option:

###### cleartool endview –server r5\_integration

1. **Ensure that the view cannot be reactivated**. Remove its tag and then unregister it. Use the Rational 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 is not necessary for a snapshot view whose view storage directory will not be moved.

1. **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 all supported operating systems:

v On Linux or the UNIX system, you can use a utility such as **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 platforms running Linux or the UNIX system. For more information, see the reference pages for your operating system.

v On Windows, you must use a copy utility that preserves file and directory ownership and access control information (see [“Preserving NTFS ACLs](#_bookmark192) [when copying a VOB or view storage directory” on page 200)](#_bookmark192) and you must run the copy utility on the host to which the view is being moved.

The following example uses the Rational 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**

v If you are moving a snapshot view on Linux or the UNIX system, 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](#_bookmark136) [moving snapshot views on Windows” on page 140.](#_bookmark136)

1. **Register the view at its new location and create a new view tag.** Use the Rational ClearCaseAdministration 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 Rational 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**

###### Change the view’s access permissions back to read/write.

**cleartool chview -readwrite -cview venus**

1. **Reactivate and test the view.** While you are still logged on as the view owner, access the view and verify the following:

v You can check out versions and, in a dynamic view, create view-private files.

v All view-private objects were moved.

v No files are displayed as hijacked (snapshot views only).

1. **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).
2. **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.
3. **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 Rational ClearCase **ccopy** utility (described in [“Preserving NTFS](#_bookmark192) [ACLs when copying a VOB or view storage directory” on page 200)](#_bookmark192) to copy the view storage directory. You must run the copy utility on the host to which the view is being moved. 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 Rational ClearCase, but does not preserve file modification times.

## Moving a view to a host running Linux or the UNIX system with a different architecture

This section documents the procedure for moving a view between hosts with different binary formats running Linux or the UNIX system.

Moving a view to a host with a different architecture that is running Linux or the UNIX system 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

1. **Stop the view’s server process**. Use the **cleartool endview** *view\_tag* command with the **–server** option:

###### cleartool endview –server r5\_integration

1. **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.

1. **Ensure that the view cannot be reactivated**. Remove its tag and then unregister it. Use the Rational 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 is not necessary for a snapshot view whose view storage directory will not be moved.

1. **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**tar** utility supported on Linux and the UNIX system, 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 platforms running Linux or the UNIX system. See the reference pages for your operating system.

1. **Register the view at its new location and create a new view tag.** Use the Rational ClearCaseAdministration Console or the following commands:

**cleartool register –view /net/venus/viewstg/v5\_integration.vws**

**cleartool mktag –view –tag v5\_integration \**

**-nstart /net/venus/viewstg/v5\_integration.vws**

###### re-create the view database from the dump files.

**cleartool reformatview –load –tag sv5\_integration**

1. **Reactivate and test the view.** While you are logged on as the view owner, access the view and verify the following:

v You can check out versions and, in a dynamic view, create view-private files.

v All view-private objects were moved.

v No files are displayed as hijacked (snapshot views only).

1. **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).
2. **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 such as **rmdir** or Windows Explorer. On the new view host, delete the backup view database (typically named db.dumped).
3. **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 Rational 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 view server host running Linux or the UNIX system 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

1. Delete the existing view tag. You can use the Rational ClearCase Administration Console or the following command.

###### cleartool rmtag –view V4.1\_Int

1. Create a new view object and tag specifying a new view server host and the existing storage. You can use the Rational 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**

1. **Reactivate the view** on the replacement view server host.

###### cleartool startview V4.1\_Int

## Displaying the properties of a view

You can use the Rational 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 Rational ClearCase Administration Console or the **–long** and **–full** options to **lsview**, you can get additional information, including:

v The view’s text mode

v The view’s protections

v Whether or not the view’s ownership and permissions are consistent

v When and by whom the config spec was last updated

v Whether the view is a dynamic view or a snapshot view

v Whether the view is read-only or writable

v For a dynamic view, whether it creates shareable or nonshareable derived objects

v When and by whom view-private data was last accessed

v When and by whom a view-private derived object was last updated

v 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 Rational 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 Rational 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 Rational 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 Rational ClearCase registry and all references to the view held by any VOBs), use the Rational ClearCaseAdministration 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.

###### 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.

1. If the view tag still exists, remove it from the registry. In the Rational 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*

1. Unregister the view. In the Rational ClearCase Administration Console, you can use the **View Object** node to remove a view object. You can also run the following command, using the view’s UUID from Step [1 on page 144:](#_bookmark140)

**cleartool unregister –view –uuid** *uuid*

1. Remove references to this view from each VOB that holds them. In the Rational 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 run the following command, using the view’s UUID from Step [1 on page 144:](#_bookmark140)

**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.

# Chapter 10. Backing up critical Rational ClearCase data

Ensuring frequent, reliable backups of essential data is a critical task for any Rational ClearCase Administrator.

Backup procedures for VOBs, views, and the Rational ClearCase registry.

## Requirements for VOB and view backup tools

VOB &view data is stored in ordinary files & dir .3 recommendations,otherwise backup will be useless:

v **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.

v **The backup tool should back up files even if they are open for writing.** On all supported operating systems, 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 Rational 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.

v **The backup tool should preserve file access times.** On some platforms running Linux or the UNIX system, 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 176)](#_bookmark171) and may prevent these pools from ever being scrubbed. The **cpio** utility, which is supported on hosts running Linux or the UNIX system, 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.

## VOB backup strategies

When planning a VOB backup strategy, you must consider several things:

v The backup tool must capture all of the necessary data.

v 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.

v A VOB may be part of a group of VOBs that are connected by hyperlinks and should be backed up and restored together.

v If UCM and IBM Rational ClearQuest are in use, a PVOB and a Rational ClearQuest database are likely to hold references to each other. These references may become incorrect when the PVOB or the Rational ClearQuest database is restored.

## Choosing between standard and snapshot backups

v A standard backup strategy that uses the tools you normally use to back up the file systems of VOB server hosts

v A snapshot backup strategy that uses the **vob\_snapshot** utility to reduce VOB lock time

1. 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.
2. If you use a backup utility that cannot capture files open for writing, Rational ClearCase must be stopped as well.

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.

When you use a standard backup procedure: (Choose backup tool, that can do open for writing)

v You can use a backup tool that cannot back up files that are open for writing, but only if you first stop Rational ClearCase on the VOB host. The time required to stop and start Rational ClearCase may not add significantly to the time it takes to back up the VOB. While Rational ClearCase is stopped, no VOBs or views on the host are accessible.

v 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.

## Backing up VOBs with standard backup procedures

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](#_bookmark145) [page 149](#_bookmark145) for details).
3. Unlock the VOB.

## Determining the location of the VOB storage directory

use the Rational 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.

## If the VOB has remote storage pools

If a VOB on a host running Linux or the UNIX system has remote storage pools, you must back them up too, regardless of the backup strategy (standard or **vob\_snapshot**) that you use.

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 specify remote locations:

**cleartool lspool –long –invob vob:/vobs/flex**

pool "cltxt01"

.

.

pool storage global pathname "/vobstore/flex.vbs/c/cltxt01"

.

.

**Note:** Many backup utilities on Linux or the UNIX system, 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 4](#_bookmark145) to determine the relative importance of the various directories.

|  |  |
| --- | --- |
| *Table 4. 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 in Rational ClearCase LT) |
| Cleartext storage pools | Optional |
| Administrative directory | Optional |

### DO pool backup

**Note:** Rational ClearCase LT does not support derived objects, so the DO pools of Rational 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:

v 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.

v 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.

v Mature projects may refer to DOs that are hard to rebuild (they may require compilers that are no longer available, or views that were deleted). These DOs are valuable, and should be backed up regularly or, better yet, versioned as elements.

Even if you do not back up entire DO pools, back 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**.

### Cleartext pool backup

Backing up cleartext storage pools is not important, because type managers re-create cleartext data containers as necessary.

Even if you do not back up entire cleartext pools, back 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 Rational 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.

## Do not perform incremental backups

Instead of modifying an existing data container when a new version of an element is checked in, Rational 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, incremental backups of VOB storage should be avoided or, if this is not possible, only a few incremental backups should be performed before the next full backup.

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 Rational ClearCase scheduled task. When you use **vob\_snapshot**:

v 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.

v 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** re-synchronizes the VOB database and the storage pools at VOB restore time.

v 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.

v 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](#_bookmark179) [with checkvob -force -fix” on page 184](#_bookmark179) 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. 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 Rational 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:

v Back up the entire VOB storage directory, including any remote storage pools (see [“If the VOB has remote storage pools” on page 149),](#_bookmark145) 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**.)

v 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. eferred deletion increases disk space requirements in a active VOB. On platforms running Linux or the UNIX system, you can run 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 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:

v 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 host running Linux or the UNIX system 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).

v Do not make 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.

**Attention:** 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 Rational 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 Rational 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 Rational ClearCase LT.) You must back up both directories.

To back up a view:

1. **Determine the location of the view storage directory.** Use the Rational 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

...

1. **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*.

###### If the view is on a host running Linux or the UNIX system, determine whether it has remote storage. You can use the 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.

1. **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 on page 151.](#_bookmark147)

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.

1. If you made the view read-only in Step [2 on page 151,](#_bookmark147) make it writable.

###### cleartool chview –readwrite r5\_integration

Properties: readwrite

1. If you renamed the view storage directory, rename it to its original name.

## Backing up Rational ClearCase registry data

Rational 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.

v On hosts running Linux or the UNIX system, the registry directory is

/var/adm/rational/clearcase/rgy.

v On Windows hosts, the registry directory is *ccase–home–dir*\var\rgy**.**

On all supported operating systems, all files in the registry directory should be backed up and restored together. In addition to backing up the rgy directory, back up the registry client list, which is not located in the rgy directory.

v On hosts running Linux or the UNIX system, the client list is stored in

/var/adm/rational/clearcase/client\_list.db.

v On Windows hosts, the client list is stored in *ccase–home–dir*\var\client\_list.db**.**

When you back up registry data, use 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:

v UCM PVOBs and the component VOBs they reference are connected by hyperlinks.

v Administrative VOB hierarchies (see [Chapter](#_bookmark93) 7) are connected by hyperlinks.

v In an environment that uses UCM and IBM Rational ClearQuest, PVOBs store references to records in Rational ClearQuest databases, and Rational ClearQuest databases store references to objects in PVOBs.

[Figure 6](#_bookmark149) illustrates these relationships in a simple configuration.

PVOB ClearQuest database

References to fields

in records

References to

VOB objects

Cross-VOB hyperlinks

Component VOB 1

Component VOB 2

Component VOB 3

*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 such as the one in [Figure 6](#_bookmark149) would consist of these steps:

1. Lock all the VOBs and the Rational ClearQuest database.
2. Back up all the VOBs and the Rational ClearQuest database.
3. Unlock all the VOBs and the Rational ClearQuest database.

In all but the simplest configurations (those with a few small VOBs and a small Rational 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](#_bookmark161) [more members of a set of related databases” on page 165.](#_bookmark161)

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 Rational 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 Rational ClearQuest database and component VOBs, which in turn simplifies the reconciliation of database skew.

Whatever your backup strategy, be sure to do the following:

v Back up all members of a related group of databases in the shortest amount of time possible.

v 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 Rational ClearCase Administration Console or **cleartool describe** to display the **AdminVOB** hyperlinks that point to or from a VOB. (As described in [“Administrative VOB](#_bookmark96) [hierarchies and global types” on page 96,](#_bookmark96) 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 Rational 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.

# Chapter 11. Restoring critical Rational ClearCase data

for restoring VOBs, views, and the Rational 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 IBM Rational ClearQuest in a UCM environment) a Rational 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.

v 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”](#_bookmark151) [on page 155](#_bookmark151) and [“Restoring a VOB from backup without vob\_restore” on page](#_bookmark155) [159](#_bookmark155) 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 161.](#_bookmark157)

v 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 163](#_bookmark159) for the procedures.

v 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”](#_bookmark159) [on page 163](#_bookmark159) for the procedures.

v If you use UCM and IBM Rational ClearQuest, a number of relationships between VOB, PVOB, and Rational 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 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:

v If the VOB’s admin directory is intact, you can use the VOB storage node of the Rational ClearCase Administration Console or the **cleartool space** command.

v 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.

1. **Stop Rational ClearCase.** This ensures that the existing VOB’s server processes are stopped.
2. **Remove the existing VOB storage directory.** Use any file system utility (**rmdir**, **rm**) or a GUI such as Windows Explorer.
3. **Restart Rational ClearCase.** Starting Rational ClearCase makes other VOBs and views on the host available.
4. **Load the backup**. Re-create 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 Linux and the UNIX system, 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.
        + v If use **cpio**, no special options are required.

On Windows, follow the procedures in [“Preserving NTFS ACLs when copying](#_bookmark192) [a VOB or view storage directory” on page 200.](#_bookmark192)

**Note:** On all supported operating systems, 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.

1. **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 199](#_bookmark191) for details.
2. **Restore remote pools.** If the VOB is on a host running Linux or the UNIX system and it 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 Step10 [on page 160.](#_bookmark156)
3. **If you restored the VOB to a new location, re-register the VOB**. You can use the ClearCase Registry node in the Rational 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**

1. **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](#_bookmark164) and the **checkvob** reference page contain more information about **checkvob**. Review this information before you run **checkvob**.

###### Attention: 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.

1. **If the VOB is replicated, run the Rational ClearCase MultiSite restorereplica command.** You must perform this processing while the VOB is still locked.
2. **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.

###### Unlock the restored VOB.

The restored VOB is now ready for use, but you may need to take two additional steps:

v If necessary, resynchronize the VOB and views. See [“To synchronize dynamic](#_bookmark160) [views” on page 164.](#_bookmark160)

v 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](#_bookmark41) [changing tags” on page 36.](#_bookmark41)

## Restoring a VOB on network-attached storage

You cannot use **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 159.](#_bookmark155) 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 have 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 on page 161,](#_bookmark157) 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 155](#_bookmark151) or [“Restoring a VOB from backup without vob\_restore” on page 159.](#_bookmark155) 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 on page 161.](#_bookmark157)

1. **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.
2. **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](#_bookmark110) [another VOB” on page 111](#_bookmark110).
3. **Remove the restored VOB** after you verify that the element you need has been relocated to the temporary VOB. Use **rmvob** or the Rational ClearCase Administration Console.

###### Re-register and re-create the tag for the original VOB.

1. **If you are using a dynamic view, mount the VOB.**
2. **Relocate the element from the temporary VOB to the original VOB.** Use

**cleartool relocate**.

1. **Verify that the element is accessible in the original VOB.**
2. **Remove the temporary VOB.** Use **rmvob** or the Rational ClearCase Administration Console.

**Note:** This procedure restores the element to the version of its directory that is selected by the view in which Step5 [on page 161](#_bookmark157) 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 Rational 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.*

1. **Remove the original view if it still exists.** Use the Rational 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 such as **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 Rational ClearCase LT), remove both the snapshot view directory and the view storage directory.

1. **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.

v On Linux and the UNIX system, 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.

v 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](#_bookmark191) [directory ACLS on NTFS” on page 199](#_bookmark191) 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.

1. **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 (Step7 [on page 163).](#_bookmark159)

1. **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 Rational 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](#_bookmark157) [on network-attached storage” on page 161](#_bookmark157) for an example.

1. **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.”](#_bookmark159)

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.

v **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 does not 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 do not 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.

v **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 Rational ClearCase Administration console or run the following command

**leartool 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 that 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:

v In a dynamic view, inspect the list of checked-out files in the **Private Files** subnode of the view storage node in the Rational ClearCase Administration Console or run **cleartool lsprivate –co**.

v In a snapshot view, run **cleartool ls –recurse –view\_only**.

v 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:

v In a dynamic view, inspect the list of checked-out files in the **Private Files** subnode of the view storage node in the Rational ClearCase Administration Console or run **cleartool lsprivate –other –long**.

v In a snapshot view, run **cleartool ls –recurse –view\_only**.

v 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 winked-in DOs that are not valid

View the list of derived objects in the **Private Files** subnode of the view storage node in the Rational 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](#_bookmark161) [consistency of a dynamic view’s derived object state.”](#_bookmark161)

## 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:

###### 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

.

.

.

1. **Remove the DOs that have no config record.** Use an operating system command such as **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 152,](#_bookmark148) 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 Rational 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:

v A component VOB is restored from backup.

v A Rational ClearQuest database is restored from backup.

v 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:

v 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).

v 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 Rational ClearQuest database

When you restore a project’s Rational ClearQuest database from backup, the following types of skew may be introduced:

v Activities defined in the PVOB may reference fields or records that do not exist in the Rational ClearQuest database.

v Any activity and stream names or titles that have been changed since the Rational ClearQuestdatabase was backed up will be different in the PVOB and the Rational 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 Rational 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 Rational ClearQuest database. The following command-line checks all entries for a project named **Release-6** in a Rational ClearQuest database named

###### CQ-UCM-DB:

**cleartool checkvob –ucm –crm\_dbname CQ-UCM-DB –project Release-6**

**Special procedures for restoring a PVOB**

A PVOB may be restored from backup by using any of the procedures documented in this chapter. Special procedures must be run after doing so, however, because restoring a PVOB may introduce skew between the PVOB and its component VOBs. Among the most common types of skew that may occur are:

v Versions, baselines, stream objects, and label types used by component VOBs may not exist in the PVOB.

v Records in the Rational ClearQuest database may reference activities that do not exist in the PVOB.

The **cleartool checkvob** command can be used to analyze and, if necessary, fix most types of skew. You can use checkvob to detect skew between the PVOB and its component VOB, and skew between the PVOB and its Rational ClearQuest database. If your PVOB is associated with multiple component VOBs in the project, then you must run checkvob for each of these component VOBs.

The **cleartool checkvob -ucm -vob\_only** command detects most types of skew between the restored PVOB and its component VOB. This command checks the PVOB for references it holds to objects in component VOBs. For example, the following command checks a component VOB whose VOB tag is \CVOB:

###### cleartool checkvob -ucm -vob\_only vob:\CVOB

The **cleartool checkvob -ucm -crm\_dbname** command detects most types of skew between the restored PVOB and the Rational ClearQuest database. For example, the following command checks for skew in a project named **Release-6** in a Rational ClearQuest database named **CQ-UCM-DB**:

###### ccleartool checkvob -ucm -crm\_dbname CQ-UCM-DB -project Release-6

For more information about using **checkvob** to analyze and fix skew between a PVOB and a component VOB, see the **checkvob** reference page.

## 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 it is preferable for you to 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:

v 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.

v 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 187](#_bookmark181) and the **checkvob** reference page**.**

v **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).

# 12. Periodic maintenance

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
* Rational ClearCase scheduler, which you can use to automate many periodic maintenance chores.

## The Rational ClearCase scheduler

The Rational 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.

v 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.

v A task is an executable program or script. A task may be run 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 Rational ClearCase runs, the scheduler is integrated with the Rational ClearCase management infrastructure and supports these functions:

v Centralized management with **cleartool** or the Rational ClearCase Administration Console

v Access control based on user and group identity

v Flexible scheduling based on time of day, period, or execution of another task

v A default set of tasks and jobs that run on every host

v 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:

v Modify the scheduler access control list (ACL), if necessary.

v Add jobs to the default daily and weekly local tasks

v Define new tasks and jobs for them to run.

## Managing the scheduler access control list

The **schedule** command 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 Windows hosts and an NIS domain on hosts

running Linux or the UNIX system. 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:** Hosts running Linux or the UNIX system 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 5](#_bookmark165) shows the access types and their implications for access to the schedule and access to the ACL itself.

|  |  |  |
| --- | --- | --- |
| *Table 5. Access types in scheduler ACL entries* | | |
| **Access type** | **Access to schedule** | **Access to ACL** |
| **Read** | Read only | Read only |
| **Change** | Read and write; can start jobs | Read only |
| **Full** | Read and write; can start jobs | Read and write |

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 Rational 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 window 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:

v A job (executable program or script) to be run when the task runs

v 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.). You can place this 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 Linux or the UNIX system, and a

.bat suffix on Windows**.** If you add your program to one of these customizable tasks, you need take no further action.

1. 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: v A unique numeric ID used by a job to refer to the task

v A unique name for the task

v The pathname to the executable program associated with the task

**Attention:** 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.

1. 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 172.](#_bookmark167) 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.

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 172.](#_bookmark167) To delete a task definition, use a text editor to edit the file task\_registry in the scheduler’s tasks directory.

## 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 host, use the Rational 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](#_bookmark164) [access control list” on page 169),](#_bookmark164) 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:

v A name and an optional description for the job.

v 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 171.](#_bookmark166)

v 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.

v The schedule on which the job runs.

v Job-related events that trigger e-mail notifications and recipients for the notifications. See [“Specifying job notifications” on page 173.](#_bookmark168)

v Whether the scheduler should delete the job after it runs.

To create a new job, use the Rational 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 window 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:

v **Sequential**. The job runs immediately after another job finishes.

v **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:

v **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 Rational ClearCase Administration Console to force immediate one-time execution of any scheduled job.

v **Run daily or every n days**. Specify the frequency of the job and the time of day the job starts.

v **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.

v **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 Rational ClearCase Administration Console:

v 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 window in which you supply the information needed to define a new job. After you specify the task, click the **Schedule** page to specify the schedule.
* To specify the schedule for an existing job, select a job and click **Action > Properties**. In the window, click the **Schedule** page 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** page of the ClearCase program in Control Panel on each host where the scheduler runs. On Linux and the UNIX system, the scheduler uses the /bin/mail program to send notifications.

To specify the notification information for a job, use the Rational 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 window, supply the information needed to define a new job. After you specify the task, click the **Settings** page 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 window, click the **Settings** page 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 Rational 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 window, 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 Rational 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 window 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 Rational 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 Rational 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 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 6](#_bookmark171) describes the function of each of these utilities.

|  |  |  |
| --- | --- | --- |
| *Table 6. 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**:

v 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.

v 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.

v 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 Rational ClearCaseAdministration Console or a **mkpool –update** command. For more information, see [“Adjusting default](#_bookmark172) [scrubbing parameters” on page 177.](#_bookmark172)

## 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 do not 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:

v **Not enough space.** The disk partitions in which VOB storage pools reside may fill up frequently. A more aggressive scrubbing strategy may be necessary.

v **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 command-line syntax supported on Linux or the UNIX system, 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.

1. **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".

###### Determine the location of the VOB storage directory. Use the lsvob

command:

**cleartool lsvob /vobs/sources**

\* /vobs/sources /net/mars/vobstg/sources.vbs

1. **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 run the task several times per day. For more information about tasks and jobs, see [“The Rational ClearCase scheduler” on page 169.](#_bookmark164)

1. **Clean up the old DO storage pool**. The **chpool** command in Step2 [on page 178](#_bookmark173) does not move existing DO data containers; it only affects where a new DO’s data container is stored. Accordingly, the old storage pool should be cleaned up:

*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.

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**

1. **Register the task** in the scheduler’s task database.
2. **Define a new job** in the scheduler that runs your task daily. Choose an appropriate starting time.
3. **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.
4. **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:

v Inconsistencies between a VOB’s database and its pools

v Improper protections on storage pools

v Improperly configured **AdminVOB** hyperlinks

v Local types that eclipse global types

v Broken cross-VOB hyperlinks, including those that link UCM components to PVOBs.

You can use **checkvob** for a variety of purposes:

v When you restore a VOB from backup media without using **vob\_restore** (the

**vob\_restore** script runs **checkvob** as part of its normal operation)

v When you restore a Rational ClearQuest database used by a UCM project

v When you move a VOB

v As part of your periodic VOB maintenance routine

## Replicated VOB considerations

**Note:** This section does not apply to Rational ClearCase LT.

The **checkvob** command is a per-replica operation. You run it to achieve local pool or database consistency. The **checkvob** command 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 Rational ClearCase 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:

v 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.

v 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 83,](#_bookmark83) 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:

v 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.

v A system or network failure prevents a Rational 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), the **checkvob** command 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.

* 1. Locks the element.
  2. Fixes missing protection problems.
  3. Fixes missing data container problems by scanning the pool for missing or misplaced containers and reconstructing containers as necessary.
  4. Updates the VOB database to reference the reconstructed containers.
  5. For missing version data, updates the VOB database to dereference lost versions with the equivalent of **rmver –data**.
  6. Applies minor events to the element’s event history.
  7. Move alternate (unreferenced) containers for this element to pool’s

lost+found directory.

* 1. 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:
   1. Fixes missing protection problems.
   2. 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 IBM support for Rational products.
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:

1. Scans the VOB database for missing (referenced, but not found) or misprotected containers.
2. Generates a list of problem VOB objects.
3. If **–fix** is specified, processes each VOB object on the list as described in [“Individual file element or DO processing” on page 180.](#_bookmark175)
4. Scans for unreferenced data containers.
5. 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:

v **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.

v **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.

v **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 are both current, or they were retrieved as a unit from a complete backup of the VOB storage directory.

## Pool name and protection problems

The **checkvob** command can detect the following problems with pools.

### Source, DO, or cleartext pool: bad pool roots

v **Description:** Misnamed or misidentified pools**.**

v **Cause: mkpool**, **rmpool**, or **rename** *pool* in the interval between database and storage pool reference times.

v **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. The **checkvob** command 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

v **Description:** Pool has an incorrect FAT RO attribute or NTFS ACL.

v **Cause:** User copied or restored pools or containers without preserving protection information.

v **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 the **checkvob** command 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** page, 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 on page 182.](#_bookmark177)

1. For all files and directories in the VOB storage directory, use the **Security** page to grant full rights to the Rational ClearCaseadministrators group and the VOB owner.
2. Log out.
3. Log on as the VOB owner.
4. Take ownership of all files and directories in the VOB storage directory.
5. 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:

v **Missing data container.** A container is considered missing if it does not are displayed as 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, The **checkvob** command attempts to fix missing containers by scanning all storage pools for alternate containers from which to reconstruct the missing ones.

v **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

v **Description:** The VOB database references a source pool data container that does not exist at the expected location.

v **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.

v **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)

v **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.

v **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.

v **Fix processing:** The **checkvob** command 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. The **checkvob** command 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. For more information, contact IBM support for Rational products.

### DO pool: Missing container

v **Description:** VOB database references a DO pool data container that does not exist at the expected location.

v **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.

v **Fix processing:** Using the equivalent of **rmdo**, **checkvob** adjusts the database to remove the reference the DO container.

### DO pool: Unreferenced container (debris)

v **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.)

v **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.

v **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.

* 1. 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.

* 1. 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.

1. If container reconstruction is incomplete, collect and report a list of missing versions.
2. If **–force** is specified or the fix? prompt is accepted, update VOB database:
   1. Adjust the database to reference reconstructed containers.
   2. Adjust the database (with the equivalent of **rmver –data)** to remove references to lost version data.
3. 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:

v A **chtype binary\_delta \*.eps** 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.

v A **chpool** operation is not reflected by older storage pools. The **checkvob** command 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:

v **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 such as **\main\br1\0** and

###### \main\br1\br2\0.

v **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, run:

**checkvob –force –fix –data –pool –source** *VOB-stg-pname*

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.

The **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 correct identity information at the correct 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:

v VOBs with multiple **AdminVOB** hyperlinks

v Global types with local copies whose names do not match

v Eclipsing types

v Eclipsing locks on local copies of global types

v 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".

1. 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:

v Eclipsing local types

v Eclipsing local locks

v Protection mismatches

v Name mismatches

1. Review the log file, and then run **checkvob –global –fix** to correct any problems. With the **–fix** option, **checkvob** does the following:

v Changes the name of each local copy to match the name of its global type.

v 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.

v 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 (**–hlinks**) mode, **checkvob** examines the hyperlinks that point to and from a VOB object itself. (It does not examine hyperlinks that point to or from objects in the VOB.) There are several predefined cross-VOB hyperlink types, including **AdminVOB** hyperlinks (see [Chapter](#_bookmark93) 7) and **RelocationVOB** hyperlinks that link a VOB to another VOB to which some of its contents have been relocated (see [“Relocating elements to another VOB” on page 11](#_bookmark110)1). You can also define your own hyperlink types and use them to connect one VOB to another.

**checkvob –hlinks** reports any cross-VOB hyperlinks that are displayed as to be incorrect. A cross-VOB hyperlink can are displayed as to be incorrect for any of the following reasons:

v The VOB at the other end of the hyperlink has no tag in the current region.

v The VOB at the other end of the hyperlink has no replica in the current registry.

v The VOB at the other end of the hyperlink has been removed.

**Note:** Do not run **checkvob –hlinks** before you run **checkvob –ucm** (see [“Using](#_bookmark182) [checkvob in a UCM environment” on page](#_bookmark182) 188). **checkvob –hlinks** may prompt you to remove hyperlinks that **checkvob –ucm** can fix.

When you run **checkvob –hlinks**, it reports problems in the following way:

###### cleartool checkvob –hlinks vob:*vob\_tag*

Unable to determine if the following hyperlink is intact. *hlinkID fromVOB*

-> *toVOB*

Delete it? [no]

where *hlinkID* is a hyperlink identifier of the form *HlinkType*@*id*@*vob-tag*, *fromVOB* identifies the VOB at which the hyperlink originates and *toVOB* identifies the VOB to which the hyperlink points. If any of hlinkID, *fromVOB*, or *toVOB* cannot be determined, it will be displayed as <unavailable>. For example:

###### cleartool checkvob -hlinks vob:\projects

Unable to determine if the following hyperlink is intact. <hlink-id not available> <object not available> -> vob:\projects Delete it? [no]

Accept the default and leave the hyperlink in place unless you are certain that checkvob has found a cross-VOB hyperlink that refers to a VOB that has been deleted. This is especially important when using Rational ClearCase MultiSite, because hyperlink removals performed by **checkvob –hlinks** are replayed at other replicas where the hyperlinks may still be valid. This can result in the removal of valid **AdminVOB** hyperlinks, which can cause problems for administrative VOB hierarchies.

## Using checkvob in a UCM environment

UCM environments, especially those that are integrated with IBM Rational ClearQuest, can create complex relationships among a project’s VOBs and its Rational 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](#_bookmark148) [related databases together” on page 152](#_bookmark148) describes ways in which you can schedule VOB and Rational ClearQuest database backups to minimize restore-time inconsistencies between them. [“Restoring one or more members of a set of related](#_bookmark161) [databases” on page 165](#_bookmark161) 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.)

## View storage maintenance for dynamic views

View storage maintenance for dynamic views includes two principal tasks:

v 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.

v Removal of unneeded view-private objects. As with 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 Rational ClearCase Administration Console displays information on disk space used by views:

v 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.

v 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.

v 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:

v The **dospace** command shows disk space used by shared derived objects in a VOB. The display shows which views have references to these DOs.

v The **space –view** command shows current and historical disk space used for a view.

The Rational ClearCase scheduler runs several jobs that gather data on disk space used by views:

v Daily data gathering on disk space used by views

v Weekly data gathering on disk space used by shared derived objects

v Daily and weekly execution of jobs that you can customize to run your own programs

For more information, see [“The Rational ClearCase scheduler” on page 169.](#_bookmark164)

## 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:

v In a view that is frequently used for builds, build scripts replace old (and potentially redundant) DO data containers with newer ones.

v 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 that the operating system supplies for removing files. To list view-private files by the

pathnames at which they are displayed as in VOBs, use the **Private Files** subnode of the view storage node in the Rational 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 do not 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.

1. **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.

1. **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**

# Chapter 13. Importing data

**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 *IBM Rational ClearCase Guide to Managing Software Projects*.

# Chapter 14. Troubleshooting

v Fixing protection problems on NTFS VOB or view storage

v Using an existing view after you relocate elements that it references

v Finding a physical container created by the MVFS for file storage

v Editing a crontab on Linux or the UNIX system, to prevent recursive traversal of the /view directory

## Repairing storage directory ACLS on NTFS

Use NTFS-formatted disks to hold 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). 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.**

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 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

ACLs for VOB and view storage directories are established by Rational ClearCase when VOBs and views are created.

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, the built-in **LocalSystem** identity (NT\_AUTHORITY\SYSTEM) is used by Rational ClearCase LT wherever the ClearCase administrators group is used by Rational ClearCase.

On a Rational ClearCase LT server, the name NT\_AUTHORITY\SYSTEM is displayed as where the name **clearcase** appears in this example. In both Rational ClearCase and Rational 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 *(The built-in identity* NT AUTHORITY\SYSTEM*is used)*

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 *(the built-in identity* NT AUTHORITY\SYSTEM*is used)*

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.

v In most cases, you can use 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.

v 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.

v 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 201,](#_bookmark193) to check the ACLs on the target directory. If **fix\_prot –l** shows any errors, follow the procedures in [“Fixing](#_bookmark195) [protection problems” on page 203](#_bookmark195) to correct them.

## Causes of protection problems

### Copying the storage directory incorrectly

If you do not use the appropriate copy utility, ACLs on the copy will not be correct. see [“Fixing protection problems”](#_bookmark195)

### 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.

A different VOB owner will be reported by Rational ClearCase 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 203.](#_bookmark195)

### Editing permissions

If you edit a file permission, for example, by using Windows Explorer or **cacls**, Rational ClearCase users will begin having access and protection problems.

To detect VOB storage directory protection problems, use **checkvob**:

**cleartool checkvob –protections –pool** *vob-stg-pname*

## Utilities for fixing protection problems

**Three utility programs for finding and fixing VOB and view storage directory protection problems are included in Rational ClearCase:**

v **vob\_sidwalk** fixes VOB storage directory protections. It can also be used to change ownership on VOB objects. For more information, see the **vob\_sidwalk** reference page.

v **fix\_prot** fixes storage directory protections for views.

v **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 Rational ClearCase administrators group. Use this option if you have changed the Rational ClearCase administrators group name or have moved a view to a new domain that has a different SID for this group.

### Examples

v To create a **.**identity directory after moving a VOB from Windows to a host running Linux or the UNIX system. 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**

v 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

v To display the protection modes associated 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:

## lsacl

*ccase–home–dir***\etc\utils\fix\_prot –chown** *owner* **–chgrp** *group* **–chmod 0666** *pname*

**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 201.](#_bookmark193)

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) Group: NT\_WEST\usersnt (Group) (non-defaulted) ACL (revision 2):

0: allowed

SID: NT\_WEST\user (Group) rights (00000000)

1: allowed

SID: NT\_WEST\tester (Group) rights (00000000)

===== *stg-pname*\groups.sd

Owner: NT\_WEST\bob (User) (non-defaulted) Group: NT\_WEST\usersnt (Group) (non-defaulted) ACL (revision 2):

Empty ACL: all access denied

1. 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.

1. If you are fixing VOB storage, log on as the VOB owner and continue.
2. If the VOB has a supplementary group list, run this command:

**cleartool protectvob –add\_group** *group-name[***,***...] vob-stg-pname*

1. To remove the cleartext containers, run this command:

**scrubber –e –k cltxt** *vob-stg-pname*

This step must precede Step [7 on page 204](#_bookmark196) because **checkvob** cannot fix cleartext containers.

1. To fix the storage pool’s protections, run this command:

**cleartool checkvob –force –fix –protections –pool** *vob-stg-pname*

*(Owner) (Primary group)*

*(Supplementary group)*

*(Supplementary group)*

*(Owner) (Primary group)*

*(No supplementary group)*

## Locating MVFS data containers

**Note:** This section does not apply to Rational 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**:

v Element cmd.c has element type **text\_file** and is currently checked out.

v Element monet.icon has element type **file** and is not currently checked out.

v File ralph\_msg is a view-private file created by saving an e-mail message to disk.

## Determining the status of Rational ClearCase files

**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 Linux and the UNIX system;

**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?

**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 Rational ClearCase Administration Console

## Where is the view?

The **lsview** command shows the location of the view storage area:

**cleartool lsview allison\_vu**

\* allison\_vu \\phobos\_users\people\arb\vw\_store\arb.vws

The **All Views** node of the Rational 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

## 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 7. Storage directories for MVFS files* | |
| **Kind of file** | **Storage directory** |
| Version (checked-in) | VOB source storage pool (s subdirectory of the VOB storage directory). |
| Checked-out version | View-private data storage (.s subdirectory of the view storage directory) |
| Unshared or nonshareable derived object | View-private data storage (.s subdirectory of the view storage directory) |
| Shared derived object | VOB derived object storage pool (d subdirectory of the VOB storage directory) |
| View-private file | View-private data storage (.s subdirectory of the view storage directory) |

The following sections show how the ***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](file:///\\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

As with 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](file:///\\sol\view_store\arb.vws\.s\00050\8000000C.00BD.ralph_msg)

### Links and directories on Linux and the UNIX system

On computers running Linux or the UNIX system, **mvfsstorage** deals with VOB and file system link and directory objects as follows:

v 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.

v 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 view root directory on Linux or the UNIX system

**Note:** This section does not apply to Rational ClearCase LT.

On any host running Linux or the UNIX system on which 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 hosts running Linux or the UNIX system run a daily **cron** script that cleans up the file system. If this script uses a **find /** command, it runs the risk of looping infinitely.

**Note:** Any command that operates on the entire Linux or 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 Rational 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 *IBM Rational ClearCase 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.

# Chapter 15. Improving client host performance

This chapter presents techniques for improving Rational ClearCase performance on the client host. Administrative responsibilities related to client host performance include the following:

v Establishing standards for client host configurations (memory, processing power, network interface characteristics, and local storage).

v Understanding MVFS cache sizes and how to adjust them.

v Understanding view caches and how to adjust them.

v Establishing sitewide defaults for cache sizes.

**hNote:** Most of the performance tuning procedures described in this chapter deal with the MVFS. They do not apply to Rational ClearCase LT clients, or to Rational ClearCase clients that run only snapshot or Web views.

## Client host configuration guidelines

Rational 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 Rational 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 Rational 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 Rational ClearCase may need to have its resources adjusted to adequately support simultaneous use of both.

## Examining and adjusting MVFS cache size

Rational 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:

v The easiest (and best) 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 any supported operating system (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** page allows you to set the scaling factor and several other cache sizes.

v You may also adjust selected values for individual caches by using **setcache**

**–mvfs** or the **MVFS Performance** page 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. Do not use this 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** page 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 221)](#_bookmark211) 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 automatically sets by Rational ClearCase up to a maximum value of 24, based on available physical memory installed on the host. If you wish to adjust the value of the MVFS scaling factor manually, however, you may do so. [Table 8](#_bookmark205) shows how the MVFS scaling factor should be adjusted based on memory size.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Table 8. How memory size affects the MVFS scaling factor* | | | | |
| **Memory size (MB)** | **Memory<24** | **24<= Memory**  **<512** | **512<= Memory**  **<3072** | **Memory >= 3072** |
| MVFS Scaling Factor | 0 | 1 | 2 | *size*/1024 |

In [Table 8,](#_bookmark205) the value of *size* excludes memory required by the operating system kernel. For *size* values greater than 3072 MB (3 GB), the MVFS scaling factor 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 MVFS scaling factor is set to 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** page, set the scaling factor value in the **Scaling factor to initialize MVFS with more memory for better performance** field. 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 9.](#_bookmark205)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Table 9. 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 (Linux and the UNIX system) | 900 | 1800 | 2700 | 900(*f*+1) |
| Cleartext Freelist Max (Windows) | 900 | 1800 | 1800 | 1800 |
| RPC Handle Cache | 5 | 10 | 15 | 5(*f*+1), *f* < 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 |

*Table 9. How the MVFS scaling factor affects individual MVFS cache sizes (continued)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MVFS cache name** | **scaling factor**  **= 0** | **scaling factor**  **= 1** | **scaling factor**  **= 2** | **scaling factor =** *f* **(***f >* **2)** |
| readdir Block Cache | 2 | 4 | 6 | 2(*f*+1), *f < 2*  6, *f* > 2 |

## Setting individual caching parameters

Although the scaling factor can be used to make MVFS cache adjustments, you can also change the sizes of individual caches. Before you take this approach, there are several considerations that need to be taken 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, in addition to a detailed knowledge of how that application uses the MVFS.

v It is possible to adversely affect MVFS performance by changing individual caching parameters.

v Using the scaling factor to change cache sizes greatly simplifies cache tuning and is effective in the majority of cases.

v 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 219.](#_bookmark210)

[Table 10](#_bookmark206) 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 10. MVFS cache information (Part 1 of 2)*

|  |  |  |  |
| --- | --- | --- | --- |
| **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** |
| 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** |

*Table 10. MVFS cache information (Part 1 of 2) (continued)*

|  |  |  |  |
| --- | --- | --- | --- |
| **getcache output field name** | **Description** | **Adjustment mechanism:** | |
| **setcache option** | **Control Panel value (Windows)** |
| 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** |
| 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. A good goal would be to maintain a hit rate of 85% or better. The sections that follow describe each cache-miss category and describe 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:

v **acregmin**: 3 seconds v **acregmax**: 60 seconds v **acdirmin**: 30 seconds v **acdirmax**: 60 seconds

To change these values, specify a mount option for the VOB in one of the following ways:

v At mount time. For example:

**cleartool mount –options acregmin=30** *vob-tag*

You must be a privileged user to do this.

v 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

re-validation 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 or 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:

v An object cache, which facilitates retrieval of objects such as versions and branches

v A directory cache, which facilitates file system directory read access

v A stat cache, which stores file attributes

v 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:

v A per-view persistent value set when the view was created (**cleartool mkview**

**–cachesize**) or modified with the **cleartool setcache –view –cachesize** command

v A per-host persistent value set with **setcache –view –host**

v A site-wide default set with **setcache –view –site**

v 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:

v To determine whether to increase the view cache

v To check the results after changing the view cache

v To analyze other view server processes

The view cache includes these types of subcaches:

v **Lookup**. Stores data used to accelerate mapping names to objects in the view.

v **Readdir.** Stores data used to accelerate read-directory operations (for example, **ls**

or **dir**) by the view.

v **Fstat.** Stores data on file attributes used by the view.

v **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 computers running Linux or the UNIX system, 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 runs:

**kill –HUP** *view\_server-process-ID*

## Analyzing the output

**getcache –view** provides this information:

v Cache type

v Percentage of the view cache being used

v Number of entries in cache

v Number of requests made

v Percentage of cache hit rates

v Amount of view cache memory being used

Here are some suggestions for analyzing this information:

v 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.

v 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.

v If you decide to increase the view cache size for a view, use the procedure described in [“Reconfiguring a view.”](#_bookmark210)

## 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:

v The value cannot be smaller than 50 KB for 32-bit platforms or 100 KB for 64-bit platforms.

v Do not specify a value larger than the amount of physical memory on the server host that you want to dedicate to this view.

v Larger cache sizes generally improve view performance, though as cache sizes approach 4 MB, the performance improvement usually becomes less noticeable.

v 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 IBM support for Rational products 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**. Use those commands whenever possible.

### Synopsis

v Determine cache status:

**mvfscache** [ *cache\_name* ]

v Control cache operation:

**mvfscache** { **–e** *cache\_list* | **–d** *cache\_list* | **–f** *cache\_list* }

v 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 or disabled status of all MVFS caches. If you do not 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 IBM support for Rational products. 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 incorrect name cache entries in the output. These are entries that were marked as being incorrect and are not used in lookups, but are retained for statistical purposes. This helps determine how often incorrect entries are replaced with new data. Incorrect name cache entries usually occur when a change in the VOB requires the MVFS to refresh its cache.

### Examples

v Determine the status of all caches.

**mvfscache** Attr: on Name: on Noent: on Rvc: on Slink: on Cto: on

v 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

v Linux and the UNIX system:

**mvfsstat** [ **–icrvVhalzAdF** ] [ **–o** *outfile*] [ *time* ] [ *count* ]

v 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 Rational ClearCase client host.

**Directory name lookup cache (dnlc).** The dnlc section reports on use of a

name-lookup cache that maps pathnames to Rational 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 (Linux and the UNIX system) 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*

*count*

Time in seconds between samples. Display deltas on each sample. If you omit this option, only the absolute values of all information are printed.

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](#_bookmark211) [221).](#_bookmark211)

**–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:

v 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.

v 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**, 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 IBM support for Rational products.

**–F** Displays statistics for mnode operations and the directory name lookup cache. Use this option only under direction from IBM support for Rational products.

# Chapter 16. Improving VOB host performance

The work of a VOB hos:t involves both 1.) read and write access to the VOB database in addition

2.) 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.

## Minimize process overhead

The most effective measures for ensuring good performance from VOB hosts are also the easiest to implement:

v **Keep nonessential processes off the VOB host.** Do not 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 Linux or the UNIX system, or a primary domain controller on Windows.

v **Keep Rational ClearCase client processes off the VOB host.** Do not use the VOB server host as an active Rational ClearCase client (a developer’s desktop system, for example).

v **Keep view\_server processes off the VOB host.** Although you may create shared views on VOB server hosts, this activity 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:**

v Use disks with **fast seek times and high rotational speeds.**

v Where practical, dedicate a disk spindle for each VOB.

v For very busy VOBs on hosts running Linux or the UNIX system, dedicate two disk spindles per VOB: one for the database and source pools, and one for the cleartext and DO pools.

v Use stripe technology (RAID 0) for improved performance.

v Use mirroring (RAID 1) if high availability is required.

**v Avoid RAID 5, unless your benchmarks show equal performance to RAID (0+1).**

**v Place VOB storage pools on a certified NAS device.**

## Add memory for disk or file system caching

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 Rational ClearCase performance significantly and produce symptoms such as these:

v Extended periods required for **scrubber** and **vob\_scrubber** execution

v Very slow **omake** or **clearmake** builds

v Clients experience RPC time-out errors

In addition to increasing the size of the disk (file system) cache, it is also important for as much as possible of this cache be reserved for VOB database pages.

Therefore, you should offload other I/O activities from the VOB server. Examples of I/O activities that you could offload from the VOB server include:

v Moving your VOB storage pools to a NAS or other NFS server.

v Keeping view\_server processes off of the VOB host.

## Modify lock manager startup options

Every VOB server host on which the lock manager is not implemented using shared memory, runs a single lock manager process. This lock manager process 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, available on certain platforms running the UNIX system, 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.

## To change lock manager startup options

Use the following procedures to change lock manager startup options on hosts running any supported operating system.

**Note:** After you change any lock manager startup option, you must stop and restart Rational 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](#_bookmark214)

**HKEY\_LOCAL\_MACHINE\SOFTWARE\Atria\ClearCase\CurrentVersion\Lock MgrCmdLine: REG\_SZ : –a almd –f** *num* **–u** *num* **–q** *num*

This registry key is preserved when a new version of Rational ClearCase is installed.

### On hosts running Linux or the UNIX system

On hosts running Linux or the UNIX system, 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](#_bookmark214) [information” on page 227.](#_bookmark214) This file should be owned by **root** and have read-only access.

This file is preserved when a new version of Rational ClearCase is installed. When the file exists, the Rational ClearCase lock manager is started 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:

v A comment begins with a **#** character and extends to the end of the line.

v White space may precede the **LOCKMGR\_OPTS** label and may precede or follow the **=** character.

v White space must precede each option (**-f**, **–u**, **–q**) and may separate the option from the specified *num* value.

v Options that are not specified revert to default values. Incorrect 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 Rational 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 hosts running Linux or the UNIX system, 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](#_bookmark213) [226).](#_bookmark213)

###### –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 Linux and the UNIX system: 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 host running Linux or the UNIX system, 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 Linux and the UNIX system: 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 Linux and the UNIX system: 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 any other uses of the device should be eliminated or minimized.

If a VOB server must share a NAS device with other uses, keep the VOB database on the server itself, and locate just the pools on the NAS device. (This configuration is supported only for hosts running Linux or the UNIX system.)

If you experience significant performance degradation when accessing VOBs whose databases are stored on a NAS device, move the VOB database to the VOB server host.

**Note:** Because of the **additional network traffic required by the CIFS (SMB) protocol,** Windows VOB server hosts are the most likely to suffer degraded performance when they locate the VOB database on a NAS device.

# Appendix A. Configuring cross-platform file system access

enable Windows computers to access the file systems of computers running Linux or the UNIX system. This capability is required when dynamic views on Windows access VOBs or dynamic views on Linux or the UNIX system.

Two types of products are supported:

v NFS client products, which run on Windows clients and use the NFS protocol to access Linux and UNIX file systems.

v SMB server products, which run on servers running Linux or the UNIX system, and use the SMB (or CIFS) protocol to provide Windows clients with access to the server’s file system.

**Note:** This appendix does not apply to Rational ClearCase LT, which does not need cross-platform file system access because it does not support dynamic views.

## NFS client products

IBM supports using these NFS client products to provide Windows computers with access to VOBs and dynamic views on Linux or the UNIX system:

v Shaffer Solutions DiskAccess

v Hummingbird® NFS Maestro™

If you use an NFS client product, you must:

v Install it on each Windows client that will access VOBs or views on servers running Linux or the UNIX system.

v Configure each installation to meet the special requirements of Rational ClearCase.

## 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 49)](#_bookmark52) so you must disable case conversion, as described in this section.

### 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** page, select **Hummingbird NFS Maestro Client**.
3. Click **Properties** to open the client configuration window.
4. On the **File Access** page, under **Filename Case**, select **Preserve Case**

## SMB server products

IBM supports two SMB server products that make Linux and UNIX file systems accessible from Windows computers:

Samba, available from [www.samba.org,](http://www.samba.org/) and

Syntax TotalNET Advanced Server (TAS) from LSI Logic Storage Systems Inc.

## Installing and configuring Samba

Samba can be downloaded from [www.samba.org.](http://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 VOB and view server running Linux or the UNIX system that you want to access from Windows.

To configure Samba for use by Rational ClearCase:

1. Create a Samba username map for the Rational ClearCase server process user.
2. Configure Samba globals.
3. Create shares for VOB and view storage.
4. Start Samba services.

### Mapping the Rational ClearCase server process user

Samba requires a username map that associates the Windows account for the Rational ClearCase server process user with an appropriate user account on Linux

or the UNIX system. (For more information about this account, see [“Common user](#_bookmark48) [and group names” on page](#_bookmark48) 45.) The examples in this section assume that the name of the Rational 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. Create this 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 user account on Linux or the UNIX system that meets the criteria listed in [“Specifying the Windows domain for](#_bookmark49) [credential mapping of users on hosts running Linux or the UNIX system” on page](#_bookmark49)

[46.](#_bookmark49) 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 Web address of this format:

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

where *computer* is the host name of a VOB or view host running Linux or the UNIX system on which you have installed Samba and *port#* represents the SWAT port number. (The default value is 901.)

1. Log on as **root.** The SWAT interface is displayed in your browser.

### Configuring Samba globals for Rational 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 11](#_bookmark224).

*Table 11. Samba global settings for Rational ClearCase*

|  |  |
| --- | --- |
| Base options | |
| **workgroup** | Set to the name of the Windows domain to which Rational 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** |
| MS-DOS attribute mapping options | |
| **map archive, map hidden, map system** | No |

Rational 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 field to the right of the **Create Share** button. To simplify administration, the share name should be similar to that of the directory on Linux or the UNIX system whose name you enter in Step [4 on](#_bookmark224) [page 239.](#_bookmark224)
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 on Linux or the UNIX system using Samba. Configure your 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.

* 1. Click **Start nmbd**. The page refreshes and should display the **nmbd** status as

running.

### Configuring Rational ClearCase to support Samba

For all Rational 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** page:

###### v 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.

v If you are using a 64-bit build of Samba 2.2.6 or later, set the value for both to 800.

v If you are using any other version of Samba, set the value for both to 200.

1. Click **OK** to apply the changes and close the window.
2. Shut down and restart Windows.

### Testing the Samba configuration on non-ClearCase files

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 user and group ownership and permissions for that file are correct on Linux or the UNIX system.
4. Verify that all Windows clients can access the Samba share, including testing permission and access restrictions.

### Testing the Samba configuration with Rational ClearCase

To verify that Rational ClearCase and Samba are working together properly:

1. On a VOB or view server running Linux or the UNIX system, install and configure Samba as described in this chapter, creating shares for VOB storage and for view storage.
2. Verify that your Rational ClearCase user and group assignments are appropriate, as described in [“Fundamentals of VOB and view access control”](#_bookmark57) [on page 55.](#_bookmark57)
3. Verify that you can access VOBs and views on the server from a client running Linux or the UNIX system.
4. Log on to a Rational ClearCase client on Windows. Use the Region Synchronizer to import VOB tags and view tags into the Windows region, for VOBs and views hosted on the server running Linux or the UNIX system.
5. Verify that you can use these views and VOBs by performing some basic Rational ClearCase operations (for example, **mkelem**, **checkin**, and **checkout**) in them.

## Installing and configuring TAS

Rational 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 platforms running Linux or the UNIX system.

This section describes how to install TAS and configure TAS and Rational ClearCase to support cross-platform file access. If you use TAS, you must install and configure it on each VOB and view server running Linux or the UNIX system, 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:** IBM supports both TAS 6.*x* and 7.*x* for use with Rational ClearCase. TAS 7.*x* includes a simplified configuration GUI, described in [“Configuring TAS 7.x](#_bookmark229) [to support Rational ClearCase” on page 244.](#_bookmark229)

### 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 Rational 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 Web address of this format in a Web browser:

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

where

v *computer* is the host name of a VOB or view host running Linux or the UNIX system, on which you have installed TAS

v *port#* represents the Framework port number (the default is 7777) The Syntax Enterprise Services page is displayed.

1. Click **Syntax Administration Framework**; a Framework logon program is displayed.
2. Log on as **root**. The Framework interface is displayed in your browser.
3. Click **TAS Configuration and Administration** in the sphere frame. The TAS configuration and administration menu is displayed 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 Rational 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 Rational 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 Rational 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 is displayed.

Configure the CIFS realm as follows:

v **Server name** — Type the name of the server, if it is not already the default.

v **Workgroup** — Type the name of the Windows domain to which your Rational ClearCase hosts and users belong.

v **Transports** — Select the protocols appropriate for your site.

v **Device for NetBEUI** — Accept the default.

v **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 Rational ClearCase server process user (TAS 6.x)** TAS requires a username map that associates the Windows account for the Rational ClearCase server process user with an appropriate user account on Linux or the UNIX system. (See [“Common user and group names” on page 45](#_bookmark48) for more information about this account.) This section assumes that the Rational 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 is displayed.
2. Click **Username Maps**; the **Username Maps** pane is displayed. Make these changes to support Rational ClearCase:

v In the field, type the name of an existing user account on Linux or the UNIX system, and click **Create**. This account must meet the criteria listed in [“Specifying the Windows domain for credential mapping of users on hosts](#_bookmark49) [running Linux or the UNIX system” on page 46](#_bookmark49)

v 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 or view storage are physically located. Clients use the volume name to represent the path to the physical VOB or view storage location.

**Note:** 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 Rational 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 field.

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 field 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 computer running Linux or the UNIX system.) If these names do not correlate, examine the volume properties to determine which pathnames are associated with which volumes.

1. Click **Create**; a **New Volume Definition** pane is displayed. Make these changes to support Rational ClearCase:

v **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.

v **Volume umask** — Type **002**.

v **Filename Case** — Select **preserve**.

v **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 Rational ClearCase:

1. Access the file service:
   1. Click **CIFS (NB) Realm** in the menu frame.
   2. Click **Manage CIFS File Services**; a list of the file services is displayed.
   3. Click the file service that corresponds to your TAS server; then click

**Administer**. A menu of file service operations is displayed.

1. Click **Configuration**; an update file service form is displayed. Make these changes to support Rational ClearCase:

v **Volume references** — Select the TAS volumes this file service references and exports.

v **Browse master** — Select **off**.

v **Umask** — Type **002**.

v **Freespace report method** — Select **root**.

v **Windows 95 logon server** — Clear this check box.

v **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.

1. Click **Authentication Options**; the **Authentication Options** form is displayed. Under **User-mode authentication options**, click **Local** or **Remote**.

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

For information about the authentication mode for your site, see your system administrator.

1. If you select **Remote** authentication, configure the authentication as follows:

v **Proxies**—Click **Proxies** and type the name of the proxy servers in this field, 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](#_bookmark227) [6.x)” on page 242.)](#_bookmark227)

v **Use Username map** — Select this check box to ensure that the file service references the **clearcase\_albd** username map specified in [“Mapping the](#_bookmark227) [Rational ClearCase server process user (TAS 6.x)” on page 242.](#_bookmark227)

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

v **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.

v **Use Username map** — Select this check box to ensure that the file service references the **clearcase\_albd** username map specified in [“Mapping the](#_bookmark227) [Rational ClearCase server process user (TAS 6.x)” on page 242.](#_bookmark227)

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 Rational ClearCase

TAS 7.*x* includes a Framework interface devoted to the configuration of a Rational ClearCase service. The **Configure ClearCase Service** page presents all TAS configuration steps that are specific to Rational ClearCase on a single screen and establishes values for the following TAS service attributes:

v A TAS service (new or existing) to be used by Rational ClearCase v Users with permissions to update Rational ClearCase information v A volume for the storage of VOB data

v A volume for the storage of view data (optional)

v 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:

v **preserve-whitespace** is set to **on**.

v **Browse master**, **null password login**, and **Windows 95** capabilities are disabled.

v The **freespace report** method is set to **root**.

v Opportunistic locks are disabled.

v Filename case is set to **preserve**.

v umask value is set to **002** (as opposed to the file service default umask **007**).

v Username maps are enabled.

**Note:** If an existing service is used for the Rational ClearCase service, the attribute values listed above override the previously configured values of the existing service.

To configure a TAS service for Rational 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:

v **Service Name** — Type the name of the new service or select an existing service to reconfigure from the list.

v **Linux or Unix ClearCase Admin Account** — Type the name of the user account on Linux or the UNIX system to which the Windows ClearCase server process user should be mapped. For more information, see [“Specifying](#_bookmark49) [the Windows domain for credential mapping of users on hosts running](#_bookmark49)

[Linux or the UNIX system” on page 46.](#_bookmark49)

v **NT ClearCase Admin Account** — Type the name of the Windows Rational ClearCase server process user account. (This manual uses the name **clearcase\_albd** for this account).

v **VOB Storage Area Path** — specify the path of the directory on Linux or the UNIX system where VOBs will be (or have been) created. To browse for this location, click **Browse**.

v **View Storage Area Path** — specify the path of the directory on Linux or the UNIX system 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.

v **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:** Select proxy authentication when using TAS with Rational ClearCase. If you are reconfiguring an existing service, the authentication method of that service is displayed for the **Authentication** attribute. In addition, if the existing service was configured for proxy

authentication, the previously configured proxy servers are displayed as 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.

1. Click **Submit** to create to reconfigure the TAS service for Rational 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.

1. 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 Rational ClearCase. You can exit the Framework Web interface.

### Configuring Rational ClearCase to support TAS

For all Rational 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** page:

###### v 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.

v Set the value for both to 800.

1. Click **OK** to apply the changes and close the window.
2. Shut down and restart Windows.

### Testing the TAS configuration on ordinary files

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).

1. Install and configure TAS as described in this chapter, using **tasstore** as the volume name and /tasstore as the path name for the volume.
2. On a Windows client, create a file in the TAS volume. Then verify that the user and group settings for that file on Linux or the UNIX system are correct.
3. 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 Rational ClearCase

To verify that Rational ClearCase and TAS are working together properly:

1. On a VOB or view server running Linux or the UNIX system, install and configure TAS as described in this chapter, creating volumes containing VOB or view storage.
2. Verify that your user and group name are valid on all hosts. (For more information, see [“Common user and group names” on page](#_bookmark48) 45.)
3. Verify that you can access VOBs and views on a server running Linux or the UNIX system from a client running Linux or the UNIX system.
4. Log on to a Rational ClearCase client on Windows. Use the Region Synchronizer to import VOB tags and view tags for VOBs and views hosted on Linux or the UNIX system into the Windows region.
5. Ensure that you can use these views and VOBs by performing some basic Rational ClearCase operations (for example, **mkelem**, **checkin**, and **checkout**) in them.

# Appendix B. Rational ClearCase and Windows domains

The operating system provides the details of user identity and group membership that determine a user’s rights to run operations on versioned artifacts in Rational ClearCase 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 required by Rational ClearCase, and about the effect of domain trust relationships on Rational 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 Rational ClearCase

Rational 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:

v Master and multimaster 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.

v Active Directory domains that are part of a single forest.

v Domain upgrade and domain migration environments that support a combination of Windows NT and Active Directory domains.

## What Rational ClearCase requires from any domain

Both Rational ClearCase and Rational ClearCase LT impose several requirements on any domain environment in which they operate:

v All users who access a common set of VOBs and views must have domain accounts and have at least one group membership in common.

v 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.

v Additional steps must be taken to enable users from multiple domains to access a common set of VOBs and views.

Rational ClearCase (but not Rational ClearCase LT) also requires a special domain account that provides a user identity for the **albd\_server** process.

**Note:** If members of a Rational ClearCase or Rational ClearCase LT community access VOBs and views on hosts running Linux or the UNIX system, user and group names for their domain accounts must be identical to those in the account database on Linux or the UNIX system. For more information, see [“Common user and group names” on page 45.](#_bookmark48)

## Rational ClearCase on nondomain hosts

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

© Copyright IBM Corp. 1992, 2005 **249**

v They cannot host VOBs or views used by other Rational ClearCase hosts.

v They cannot access VOBs and views hosted on other Windows computers.

Nondomain systems can function as Rational ClearCase clients in networks in which all VOB and view storage resides on hosts running Linux or the UNIX system.

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 Rational ClearCase.

## Domain user and group accounts

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

The Rational ClearCase users group must have the following characteristics:

v It must be a domain global group or an Active Directory universal group. Use a domain global group unless the group needs to include other groups.

v If members of the group must access VOBs or views on a host running Linux or the UNIX system, the group name must be the same as the name of the ClearCase users group on Linux or the UNIX system. For more information, see [“Common user and group names” on page 45.](#_bookmark48)

**Note:** If your Rational 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 Rational ClearCase users group for each of these groups.

In addition to the Rational ClearCase users group, a Rational ClearCase community (but not a Rational ClearCase LT community) requires two additional domain accounts:

v A Rational ClearCase server process user. The **albd\_server** program runs with this identity. In examples throughout this document, the Rational ClearCase server process user is named **clearcase\_albd**. The Rational ClearCase server process user must be a member of the Rational ClearCase administrators group.

v A Rational ClearCase administrators group. Members of this group can perform all Rational ClearCase operations, including those that permanently destroy data. Membership in this group should be restricted to the Rational ClearCase server process user and a few Rational ClearCase administrators.

**Note:** The Rational ClearCase server process user and Rational ClearCase administrators group are not required by Rational ClearCase LT, which uses the built-in **LocalSystem** account for server processes and grants any administrator of the Rational ClearCase LT server administrative privileges to Rational ClearCase.

The Rational ClearCase server process user and Rational 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](#_bookmark233) [domain accounts manually” on page 251.](#_bookmark233)

## Setting the Rational 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).

Rational 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 Rational ClearCase users group (for example, **NT\_WEST\clearusers**). This guarantees an unambiguous definition of the group that Rational ClearCase considers the user’s primary group. Rational 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 Rational ClearCase users group who are also members of the Rational ClearCase administrators group must set CLEARCASE\_PRIMARY\_GROUP to the name of the Rational ClearCase users group, not the name of the Rational ClearCase administrators group.

To verify that CLEARCASE\_PRIMARY\_GROUP has been properly set:

###### Click Start > Programs > Rational Software > Rational ClearCase > Administration > ClearCase Doctor.

1. Click **Start Analysis**.
2. When the analysis is finished, click the **Topics** page and open the **User Login Account** folder.
3. 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 Rational ClearCase LT.

The Rational ClearCase Site Preparation wizard attempts to create domain accounts for the Rational ClearCase administrators group and Rational 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 Rational ClearCase installation program on individual hosts.

If the user running the Rational 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** field, type the following text:

**Used exclusively by ClearCase server processes and administrators.**

**Note:** In Active Directory, the type of the group must be **security** (not

###### distribution).

1. Create a new user, **clearcase\_albd** (or another user name the community has chosen), and put the group name defined in Step [2 on page 251](#_bookmark233) in the new user’s group list.

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

v In the **Description** field, type the following text:

###### Used exclusively by ClearCase servers

v On each Rational ClearCase host that is configured to support local VOBs and views, the **clearcase\_albd** user must have rights to **Log on as a service**. In addition, this user (as with any Rational ClearCase user) must have rights to **Access this computer from the network** or be a member of a group that has these rights. Do not give this user right to **Log on locally** unless there is a specific need to do so (such as having to change the default ACL on the Rational ClearCase job scheduler).

## Multiple user account domain support

If your existing domain configuration requires it, you can enable Rational ClearCase access for users and computers in multiple domains.

## Using active directory universal groups

When a Rational 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 Rational 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 Rational ClearCase users group as an Active Directory universal group.
3. Make each domain global group whose members are part of the Rational ClearCase community a member of the Rational ClearCase users group. Do not add 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 Rational ClearCase users to set CLEARCASE\_PRIMARY\_GROUP to the domain-qualified name of the (universal) Rational 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](#_bookmark239) [257](#_bookmark239) 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 Rational ClearCase community includes users from multiple Windows NT domains, you must enable the Rational 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, you might 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 Rational 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 Rational ClearCase in this environment, create proxy groups and enable domain mapping as follows:

1. Ensure that each Rational ClearCase host is a member of a resource domain that trusts the ATLANTA, BOSTON, and CHICAGO domains.
2. Create the Rational 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 Rational ClearCase host in each of these domains to log on as the **clearcase\_albd** user in the primary Rational ClearCase domain (in this case, ATLANTA**\clearcase\_albd**).
4. Create two more domain global groups, one in each of the other domains:

v In the BOSTON domain, create the group BOSTON**\clearusers\_Boston**.

v 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, the group is recognized by Rational ClearCase 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, members of a proxy group are treated by Rational ClearCase 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).**

1. Make Rational ClearCase users members of the appropriate domain groups:

v Make users whose accounts are in domain ATLANTA members of

###### ATLANTA\clearusers.

v 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**.

1. Enable domain mapping on each Rational ClearCase host. To do so, edit the Windows registry on that host to make the following changes:
   1. Using a Windows registry editor, navigate to HKEY\_LOCAL\_MACHINE\SOFTWARE**\Atria\ClearCase\CurrentVersion**.
   2. Click **Edit > Add Value**.
   3. In the **Add Value** window, enter **DomainMappingEnabled** as the **Value Name** and select **REG\_DWORD** as the value type.
   4. Click **OK** to start the DWORD editor
   5. In the DWORD editor, enter **1** (hex) in the **Data** field.
   6. Click **OK** to add the value.
2. Require each Rational ClearCase user to set the user environment variable CLEARCASE\_PRIMARY\_GROUP to the value **ATLANTA\clearusers**. See [“Setting the Rational ClearCase primary group” on page 251.](#_bookmark233)

**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 such as the Rational 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**, in addition to the Rational ClearCase administrators group.

## Conversion to Active Directory

As with any enterprise-scale application in a Windows network, Rational 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 Rational ClearCase and Rational ClearCase LT, and how to manage users, groups, hosts, and data during and after the conversion.

**Note:** If you use Rational ClearCase or Rational 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 the Microsoft 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 Rational 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 Rational 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 Rational ClearCase control.

Conversion to Active Directory affects Rational ClearCase in several ways:

v 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 Rational ClearCase communities in which users from multiple domains access a common set of VOBs and views.

v Windows Security Identifiers (SIDs) for users and groups can change in some conversion scenarios. Because Rational 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 Rational 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: v 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.

v 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.

A knowledgeable Rational ClearCase administrator who has reviewed this chapter and applicable documents from Microsoft and who understands the impact of various conversion or migration strategies on Rational ClearCase, should review (and if possible help plan) your organization’s conversion from Windows NT domains to Active Directory.

## Preparing Rational ClearCase hosts

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

v 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. For information on VOB schema versions, and how to upgrade VOBs from an older schema version to schema version 54, refer to [“VOB schema versions” on](#_bookmark72) [page 72.](#_bookmark72)

v The user environment variable CLEARCASE\_PRIMARY\_GROUP must be defined for all users. Set the value of this variable to a domain-qualified group name of the form *DOMAIN\_NAME\group\_name*.

Before you proceed with the conversion to Active Directory, verify that all Rational ClearCase hosts have been configured as described in this section and that Rational 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 258.](#_bookmark240)

**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 Rational ClearCase users, groups, and hosts are members of a single Windows NT domain:

1. Prepare Rational ClearCase hosts as described in [“Preparing Rational ClearCase](#_bookmark237) [hosts” on page 255.](#_bookmark237)
2. If a backup domain controller is not online during the upgrade, stop Rational ClearCase on all server hosts to prevent Rational ClearCase operations from being run 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 Rational ClearCase hosts.

## Upgrading a master domain and its resource domains

Use this procedure if all Rational 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 Rational 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](#_bookmark238) [page 256.](#_bookmark238)
2. Upgrade each resource domain, as described in [“Upgrading a single domain”](#_bookmark238) [on page 256.](#_bookmark238) 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 Rational ClearCase hosts in that resource domain.

## Upgrading multiple master and resource domains

Use this procedure when Rational 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.

Convert the Active Directory domains to native mode after the upgrade is complete, 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 Rational ClearCase hosts, as described in [“Preparing Rational](#_bookmark237) [ClearCase hosts” on page 255.](#_bookmark237)
2. If you do not have a backup domain controller online for each of the master and resource domains during the upgrade, stop Rational ClearCase on all server hosts to prevent Rational 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 Rational 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. Make all upgraded domains members of the same forest. This allows you to use Active Directory universal groups, as described in [“Using active directory universal groups” on page 252.](#_bookmark234)
6. After the upgrade of a resource domain is complete, shut down and restart all Rational 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](#_bookmark239) [groups” on page 257](#_bookmark239) 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 Rational 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 Rational ClearCase users group and all the proxy groups must exist in the same forest.

1. Use Active Directory management tools to rename the Rational ClearCase users group in the primary Rational ClearCase domain. For example, the **ATLANTA\clearusers** group created in the procedure in [“Using proxy groups](#_bookmark235) [and domain mapping in Windows NT domains” on page 253](#_bookmark235) could be renamed to **ATLANTA\clearusers\_Atlanta**.
2. Use Active Directory management tools to create a new (universal) Rational ClearCase users group. 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 on page 257](#_bookmark239) as members of the new (universal) Rational ClearCase users group.
4. Disable domain mapping on all Rational ClearCase client and server hosts. To disable domain mapping on a Rational 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 Rational ClearCase hosts may store this value in HKEY\_LOCAL\_MACHINE\SOFTWARE**\Atria\ClearCase\CurrentVersion**.)

1. Shut down and restart all Rational 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 Rational 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 Rational ClearCase:

v After a Rational ClearCase host has become a member of an Active Directory domain, you must reconfigure the host’s **albd\_server** to log as the Rational 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.

v VOB storage directories must be reprotected so that they include the new SIDs of the VOB owner in the directory ACL.

v VOB databases must be updated with the new SIDs. The Rational 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](#_bookmark243) [groups” on page 261](#_bookmark243) 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 Rational ClearCase hosts as described in [“Preparing Rational ClearCase hosts” on page 255.](#_bookmark237) Even though the hosts may not be migrated immediately, they will not be accessible by migrated users and groups until you have taken this step.

### Migrating users and groups

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 Rational 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

Rational ClearCase users group and, if they exist, the Rational ClearCase administrators group and **clearcase\_albd** account in the migration.

1. 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:

v Add the domain-qualified name of the Rational 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**

v 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 Rational 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 S-1-5-21-103034363-981818062-1465874335-1064

In this case, the user should set CLEARCASE\_PRIMARY\_GROUP to the value **S-1-5-21-103034363-981818062-1465874335-1064**

1. 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 Rational ClearCase users group in the Windows NT domain, as well as the name of the Rational ClearCase users group in the Active Directory domain. For example:

###### CLEARCASE\_GROUPS=AD-DOMAIN\clearusers;S-1-5-21-103034363- 981818062-1465874335-1064

Step [2 on page 259](#_bookmark241) 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 56.](#_bookmark58)

### If you must add a new user while migration is in progress

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

v 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 Rational ClearCase users group that has been cloned and exists in the Active Directory domain.

v 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 Rational

ClearCase hosts to the Active Directory domain. If you cannot migrate all your Rational ClearCase hosts at the same time, then migrate 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 performing any of these procedures, review the **vob\_sidwalk**

reference page to better understand the capabilities of **vob\_sidwalk**.

To migrate an individual Rational ClearCase host:

1. Stop Rational ClearCase on the host.
2. Use the procedures defined by Microsoft to migrate the host to the Active Directory domain.
3. For Rational ClearCase hosts only (does apply to Rational ClearCase LT):

v After you migrate the host, reconfigure its **albd\_server** to log on as the (migrated) Rational ClearCase server process user in the Active Directory domain. You can do this by reinstalling Rational 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 251.](#_bookmark233)

v If the host is a VOB server that will be also accessed by clients running Linux or the UNIX system, reset the credentials mapping domain in the **Options** page of the ClearCase program in Control Panel. The credentials mapping domain must be one in which user and group account names match those of users on Linux or the UNIX system that will access VOBs on this server.

1. Restart Rational ClearCase on the host.
2. 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*

In this example, *storage-dir-pname* is the pathname to the VOB or view storage directory.

1. Update VOB databases with the new SIDs that represent the cloned user and group accounts.
   1. Log on to the VOB server host as the VOB owner or privileged user.
   2. Lock the VOB for all users except yourself (**–nusers** *you*).
   3. 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](#_bookmark243) [vob\_sidwalk to change or update VOB users and groups” on page 261.](#_bookmark243)

* 1. Unlock the VOB.

**Note:** If you had been using proxy groups to enable users from multiple domains to access a common set of Rational ClearCase artifacts, an Active Directory universal group should be used to eliminate the need for proxy groups and domain mapping. Follow the procedure in [“Converting proxy groups” on](#_bookmark239) [page 257](#_bookmark239)

### 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 on page 261.)](#_bookmark243)

**vob\_siddump –sidhistory** *vob-tag SIDfile-path*

1. Log on to the VOB server that hosts the VOB whose tag you used in Step [3 on](#_bookmark243) [page 261.](#_bookmark243) 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 on page](#_bookmark243) [261](#_bookmark243) 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](#_bookmark243) [vob\_sidwalk to change or update VOB users and groups” on page 261.](#_bookmark243)

1. Unlock the VOB.

**Note:** Unless the VOB remains locked from the time you begin Step [3 on page 261](#_bookmark243) until the time you complete Step [4 on page 261,](#_bookmark243) 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 not valid, 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:

v Migrating a VOB from a Windows NT domain to an Active Directory domain v Moving a VOB to a host in a domain that does not trust the original domain v Moving a VOB from a Windows host to a host running Linux or the UNIX

system, or vice versa

v 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.](#_bookmark110) The **vob\_sidwalk** reference page provides complete information on all **vob\_sidwalk** and **vob\_siddump** options.

Regardless of the procedure you use, **vob\_siddump** (or **vob\_sidwalk**) should be run without the **–execute** option. The output should then be examined 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 such as 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 on page 262](#_bookmark244) to find its SID.
3. If the SID returned in Step [2 on page 262](#_bookmark244) is different from the SID used in Step [1 on page 262,](#_bookmark244) the SID used in Step [1 on page 262](#_bookmark244) is assumed to be a historical SID, and the SID returned in Step [2 on page 262](#_bookmark244) 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”](#_bookmark115) [on page 116](#_bookmark115) 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 such as 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

IBM Rational ClearCase MultiSite customers who use identity-preserving 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 identity-preserving 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. Remove 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 Rational 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 259](#_bookmark241) to migrate hosts. All identity-preserving 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 identity-preserving 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 on page 263](#_bookmark245) 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 on page 263.](#_bookmark245)

1. After all identity-preserving and permissions-preserving replicas have been updated (Step [2 on page 263)](#_bookmark245) and the SID file generated (Step [3 on page 263)](#_bookmark245) 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.

# Appendix C. Configuring non-ClearCase access on Linux or the UNIX system

Some Rational 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 Rational ClearCase host running Linux or the UNIX system to provide this support.

**Note:** Non-ClearCase access is not supported on hosts running Linux or the UNIX system and Rational ClearCase LT, or on any Rational ClearCase Windows host.

## Using non-ClearCase access on hosts running Linux or the UNIX system

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

v A host running Linux or the UNIX system and Rational 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.

v 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:

v **VOB access.** Users on the non-ClearCase host can read, but cannot modify, data from VOBs on VOB server hosts running Linux or the UNIX system that have been 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.

v **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 *IBM Rational ClearCase Guide to Building Software*.)

v 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 and VOB combination is as follows:

1. A Rational 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**.

© Copyright IBM Corp. 1992, 2005 **265**

1. The host starts an export view, through which the VOB is accessed by non-ClearCase hosts. An export view must meet two requirements:

v It must be marked for export by using the **–ncaexported** option to **mkview**

or **mktag**.

v If the view has tags in multiple registry regions, the exported tag must be in the view server host’s region.

1. 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.

1. 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 do not 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 such as 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 and view pair, Rational 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:

v The host on which the VOB resides

v The host on which the export view resides

v 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 Rational 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 vendor hosts. Such situations may result in time-outs (if VOBs are soft-mounted) or deadlocks (if VOBs are hard-mounted).

Avoid multihop configurations when using non-ClearCase access.

### Avoiding multihop configurations

You can avoid multihop configurations by adhering to the following guidelines:

v Locate the export view on the same host as VOB it exports. You should not run a **view\_server** and **vob\_server** on the same host (see [“Minimize process](#_bookmark212) [overhead” on page 225),](#_bookmark212) but doing so may be preferable to using a multihop configuration.

v 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, Rational 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:

v Creation of home directories

v Remote log-ons from a Rational ClearCase host

v 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, a **–access** option should be used in each entry in

/etc/exports.mvfs. This restricts the export to specified non-ClearCase hosts and 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 Rational ClearCase view and VOB pair has been exported from a Rational 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 and 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 and 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 and 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 and VOB pair.

For more information about Rational ClearCase and the automounter, see [“Rational](#_bookmark20) [ClearCase and the NFS automounter” on page 15.](#_bookmark20)

## 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:

v 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.)

v 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**, edit the mount options to request NFS V2.

v 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.)

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 platforms running Linux or the UNIX system, 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 Rational ClearCase Web interface to access VOB data from hosts that do not run Rational ClearCase.

# Appendix D. Estimating VOB size

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 Linux or the UNIX system, the pools can be located on a different partition or host, using symbolic links, as described in [“Creating remote storage pools on hosts](#_bookmark90) [running Linux or the UNIX system” on page 90.](#_bookmark90)

## 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](#_bookmark74) [guidelines” on page 74](#_bookmark74) include a recommendation that the VOB server be configured with enough physical memory to hold half of each VOB database it hosts.

[Table 12](#_bookmark251) provides information about the sizes, in bytes, of the four most significant contributors to VOB database space consumption.

|  |  |
| --- | --- |
| *Table 12. Sizes of common VOB data types* | |
| **Data type** | **VOB database space consumption (in bytes)** |
| Element with one version | 3970 |
| Each additional version of an element | 642 |
| Unshared DO (excluding config rec) | 1924 |
| Shared DO (excluding config rec) | 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](#_bookmark184) 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 Rational ClearCase LT, or use Rational 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 12](#_bookmark251) to compute a rough approximation of the VOB’s database size by using the formula:

*S* = *N* (*E* + (*e* \* (*V* -1))+ *(D* \* *d*))

© Copyright IBM Corp. 1992, 2005 **271**

where:

v *S* is the database size in bytes.

v *N* is the number of elements in the VOB.

v *E* is the size of an element with one version.

v *e* is the size of each additional version of an element.

v *V* is the average number of versions per element

v *D* is the number of unshared derived objects per element

v *d* is the size of an unshared derived object.

The following example uses this formula to estimate the size of a VOB that contains 500 elements with an average of 20 versions and 5 derived objects per element:

500 \* (3970 + (642 \* (20 - 1)) + (5 \* 1924)) = 12,894,000 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 13](#_bookmark252) lists compression ratios for commonly used built-in type managers.

|  |  |  |
| --- | --- | --- |
| *Table 13. 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 13,](#_bookmark252) you could compute a rough approximation of VOB source pool size by using the formula:

*P = N \* V \* S / C*

where:

v *P* is the source pool size in bytes.

v *N* is the number of elements in the VOB.

v *V* is the average number of versions per element

v *S* is the average file size of each cleartext element version.

v *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, there is no 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](#_bookmark84) [84.](#_bookmark84)

# Index

**Special characters**

/Windows interoperation

[and access to ClearCase objects 46](#_bookmark49) [credentials 46](#_bookmark49)

# A

access control for ClearCase objects [access to objects in VOB 57](#_bookmark59) [derived objects 68](#_bookmark70)

[dynamic views 64](#_bookmark66)

[elements 60](#_bookmark62)

[global types, changing protection 104](#_bookmark104) [groups and VOB users 82](#_bookmark82)

[how it works 55](#_bookmark57)

[in mixed OS environments 45](#_bookmark48) [locking 63](#_bookmark65)

[metadata 62](#_bookmark64)

[pools 62](#_bookmark64)

[remote root user 56](#_bookmark58) [scheduler tasks and jobs 169](#_bookmark164) [user process credentials 57](#_bookmark59) [users and groups in 55](#_bookmark57)

[view objects 64](#_bookmark66)

[view-private files 66](#_bookmark68)

[VOB object 59](#_bookmark61)

access control for file system objects [about 68](#_bookmark70)

[common protection problems 201](#_bookmark193) [fixing protection problems 203](#_bookmark195)

[admin\_server process 8](#_bookmark13) administrative VOB hierarchies

[about 96](#_bookmark96)

[adding VOBs to 97](#_bookmark97) [backing up 99,](#_bookmark99) [152](#_bookmark148) [linking VOBs to 80](#_bookmark80) [removing VOBs 100,](#_bookmark100) [101](#_bookmark101)

[replicated 106](#_bookmark106)

[restoring VOBs 167](#_bookmark163)

[restrictions 99](#_bookmark99)

[structure 97](#_bookmark97)

[AdminVOB hyperlinks 96,](#_bookmark96) [98,](#_bookmark98) [100](#_bookmark100)

[albd\_server process 7,](#_bookmark12) [8](#_bookmark13) automounter

[and ClearCase 15](#_bookmark20)

[and non-ClearCase access 267](#_bookmark248)

backups *(continued)*

[VOB, remote storage pools 149](#_bookmark145) [VOB, strategies for 145](#_bookmark141)

[build auditing, in MVFS 134](#_bookmark131)

# C

case-sensitive lookup [configuring MVFS 49](#_bookmark52)

[issues in mixed OS environment 49](#_bookmark52) [ccase-home-dir directory xv](#_bookmark2)

checkvob command [about 179](#_bookmark174)

[considerations on replicas 179](#_bookmark174) [fixing broken hyperlinks 187](#_bookmark181) [fixing global type problems 186](#_bookmark180) [how it works 180,](#_bookmark175) [181](#_bookmark176)

[use in UCM environments 188](#_bookmark182) [ClearCase administrators group 47](#_bookmark50) [ClearCase File Service (CCFS) 47](#_bookmark50) ClearCase registry

[about 25](#_bookmark30)

[administration guidelines 39](#_bookmark44) [and site default values 31](#_bookmark36) [backing up 40,](#_bookmark45) [152](#_bookmark148)

[client list 33](#_bookmark38)

[creating tags in new region 38](#_bookmark43) [data model 25](#_bookmark30)

[host assignments 34](#_bookmark39) [modifying objects and tags 34](#_bookmark39) [regions 28,](#_bookmark33) [29](#_bookmark34)

[registry password 32,](#_bookmark37) [33](#_bookmark38) [renaming server host 39](#_bookmark44)

ClearCase registry regions [adding 37](#_bookmark42)

[removing 38](#_bookmark43)

ClearCase registry server host [about 31](#_bookmark36)

[changing assigned 34](#_bookmark39)

[changing primary 40](#_bookmark45)

[defining 32](#_bookmark37)

[defining backup 40](#_bookmark45)

[promoting backup to primary 40,](#_bookmark45) [41](#_bookmark46)

[restoring primary 42](#_bookmark47)

[restoring without backup host 42](#_bookmark47) ClearCase server process user

[about 46](#_bookmark49)

[group membership 56](#_bookmark58)

|  |  |  |  |
| --- | --- | --- | --- |
| **B**  backups |  | [Samba mapping for 237](#_bookmark222)  [TAS mapping for 242](#_bookmark227) [UNIX identity for 46](#_bookmark49) |  |
| [about 145](#_bookmark141) |  | ClearCase server processes |
| [administrative VOBs 99](#_bookmark99)  [ClearCase registry data 40,](#_bookmark45) [152](#_bookmark148) |  | [about 6](#_bookmark11)  [error logs 10](#_bookmark15) |
| [DO pools 150](#_bookmark146)  [importance of locking VOBs for](#_bookmark144) | [148](#_bookmark144) | [starting and stopping 11](#_bookmark16)  [ClearCase LT, compared with ClearCase](#_bookmark7) | [2](#_bookmark7) |
| [tools requirements 145](#_bookmark141)  [types of for VOBs 146](#_bookmark142) |  | [clearexport\_pvcs utility 196](#_bookmark189)  [clearexport\_rcs utility 195](#_bookmark188) |  |
| [views 151](#_bookmark147) |  | [clearexport\_ssafe utility 193](#_bookmark186) |  |

[vob\_snapshot procedure 147](#_bookmark143) [VOB, and related databases 152](#_bookmark148)

[clearimport utility 194,](#_bookmark187) [196,](#_bookmark189) [197](#_bookmark190)

[clearlicense utility 18](#_bookmark23)

© Copyright IBM Corp. 1992, 2005 **303**

[ClearQuest databases, restoring from backup 166](#_bookmark162) cleartext pools

[about 85](#_bookmark85)

[backing up 150](#_bookmark146) [pattern of use 91](#_bookmark91)

client hosts

[configuration guidelines 211](#_bookmark203)

[with multiple network interfaces 14](#_bookmark19) component VOBs

[about 72](#_bookmark72)

[access control for objects 62](#_bookmark64) [restoring from backup 166](#_bookmark162)

configuration records [about 94](#_bookmark94)

[view storage 131](#_bookmark128) [conventions, typographical xv](#_bookmark2) [CPU capacity, of VOB host 75](#_bookmark75) [cquest-home-dir directory xv](#_bookmark2)

[Credentials Manager service 12,](#_bookmark17) [133](#_bookmark130)

[credmap\_server process 8](#_bookmark13) [customer support xviii](#_bookmark4)

# D

[db\_server process 9](#_bookmark14)

[DCHP guidelines 14](#_bookmark19) DO pools

[about 85](#_bookmark85)

[container protection on Windows 182](#_bookmark177) [missing container 184](#_bookmark179)

[patterns of use 91](#_bookmark91)

[patterns of use and backups 150](#_bookmark146) [unreferenced container 184](#_bookmark179)

DOs (derived objects) [access control 62,](#_bookmark64) [68](#_bookmark70)

[how managed in dynamic view 132](#_bookmark129) [in restored view 165](#_bookmark161)

[nonshareable 131](#_bookmark128)

[scrubbing 177](#_bookmark172)

[sharable 94](#_bookmark94)

[unshared 131](#_bookmark128) dynamic views

[about 129](#_bookmark126)

[access control 64](#_bookmark66)

[access to storage directories 39](#_bookmark44) [managing DOs in 132](#_bookmark129) [restored, cleaning up 164,](#_bookmark160) [165](#_bookmark161)

[version selection 132](#_bookmark129)

[view database 132](#_bookmark129)

[view-private storage 131](#_bookmark128)

# E

elements

[access control 60](#_bookmark62)

[changing type 96](#_bookmark96)

[relocating to another VOB 111](#_bookmark110) [restoring from backup 161](#_bookmark157) [stranded 89](#_bookmark89)

[error logging, server processes 10](#_bookmark15) [event records 94](#_bookmark94)

# F

[feature levels 73](#_bookmark73) [feature levels, VOB 72](#_bookmark72)

[fix\_prot utility 202](#_bookmark194)

# G

global types [advantages of 96](#_bookmark96)

[and -acquire option 101](#_bookmark101) [and checkvob 186](#_bookmark180)

[and eclipsing local types 101](#_bookmark101) [changing protection 104](#_bookmark104) [converting local to 105](#_bookmark105) [copying to VOBs 104](#_bookmark104)

[correcting inconsistent state of 106](#_bookmark106) [creating 101](#_bookmark101)

[history information 103](#_bookmark103)

[listing 103](#_bookmark103)

[locking and unlocking 104](#_bookmark104) [mastership 108,](#_bookmark108) [109](#_bookmark109)

[properties of 102](#_bookmark102)

[removing 106](#_bookmark106)

[renaming 105](#_bookmark105)

[group administration 6](#_bookmark11)

[group names on UNIX and Windows 45](#_bookmark48) [group, primary 55](#_bookmark57)

# H

hosts

[changing registry assignments of 34](#_bookmark39) [ClearCase 3](#_bookmark8)

[ClearCase license server 3](#_bookmark8) [ClearCase registry server 3](#_bookmark8) [ClearCase LT 2](#_bookmark7)

[configuring for non-ClearCase access 265](#_bookmark246) [release area 2](#_bookmark7)

[with multiple network interfaces 14](#_bookmark19) [hyperlinks, fixing broken 187](#_bookmark181)

# I

importing data into VOBs [about 191](#_bookmark184)

[from PVCS 196](#_bookmark189)

[from RCS 195](#_bookmark188)

[from Visual SourceSafe 191](#_bookmark184) [insert\_cr text mode 51](#_bookmark54)

# L

license database

[contents and format 21](#_bookmark26) [location 18](#_bookmark23)

license server hosts

[adding new licenses 19](#_bookmark24) [additional, setting up 19](#_bookmark24) [how specified 18](#_bookmark23)

[moving licenses 20](#_bookmark25)

[renaming 21](#_bookmark26)

[setting up 18](#_bookmark23) licenses (ClearCase LT)

[about 22](#_bookmark27)

[expiration 23](#_bookmark28)

[License Key Administrator 24](#_bookmark29) [Rational Suite license 23](#_bookmark28) [types of 23](#_bookmark28)

licenses (ClearCase LT) *(continued)*

[UNIX client 24](#_bookmark29)

[Windows client 23](#_bookmark28) licenses (ClearCase)

[about 17](#_bookmark22)

[audits of user activity 18,](#_bookmark23) [22](#_bookmark27)

[excluded users 21](#_bookmark26)

[expiration 22](#_bookmark27)

[license key 21](#_bookmark26)

[user priority 18,](#_bookmark23) [21](#_bookmark26) lockmgr process

[about 9](#_bookmark14)

[startup options and VOB host performance 226](#_bookmark213) locks

[on global types 104](#_bookmark104) [on VOBs 82,](#_bookmark82) [83](#_bookmark83)

[lost+found directory 89,](#_bookmark89) [131](#_bookmark128)

[lsacl command 203](#_bookmark195)

# M

[mastership and global types 108,](#_bookmark108) [109](#_bookmark109) metadata

[about 71](#_bookmark71)

[access control 62](#_bookmark64) moving views

NAS devices *(continued)*

[restoring VOB backup to 161](#_bookmark157) [server storage locations on 77](#_bookmark77)

[NFS client products 231](#_bookmark216)

[non-ClearCase access, configuring 265](#_bookmark246)

# O

objects (VOB) [about 26,](#_bookmark31) [93](#_bookmark93)

[access control 59](#_bookmark61) objects in VOB

[*See also* elements, global types, type objects](#_bookmark6) [about 93](#_bookmark93)

[access control for 57](#_bookmark59) [locking 63](#_bookmark65)

# P

port assignments [albd\_server process 8](#_bookmark13)

[Rational Web Platform 276](#_bookmark254) [privileged users and groups 56](#_bookmark58) [PVCS data, importing 196](#_bookmark189)

|  |  |  |
| --- | --- | --- |
| [about 138](#_bookmark135)  [architecture of new host 138,](#_bookmark135) [141](#_bookmark137) |  | **R** |
| moving VOBs |  | [Rational License Key Administrator 24](#_bookmark29) |
| [about 114](#_bookmark113) |  | [Rational Web Platform (RWP) 276](#_bookmark254) |
| [and Windows domains 116](#_bookmark115) |  | [hosts 3](#_bookmark8) |
| [precautions 115](#_bookmark114) |  | [proxy access to 280](#_bookmark258) |
| [to different operating system 122](#_bookmark120) |  | [RCS data, importing 195](#_bookmark188) |
| [to NAS devices 128](#_bookmark125) |  | registry |
| [UNIX to Windows 125](#_bookmark123) |  | [about 4](#_bookmark9) |
| [Windows to UNIX 123](#_bookmark121) |  | [release area 2](#_bookmark7) |
| [with remote storage pools 119](#_bookmark117) |  | [relocate command, how it works 111](#_bookmark110) |
| [multibyte characters, in tag names 30](#_bookmark35) |  | relocating elements |
| MVFS (multiversion file system) |  | [about 111](#_bookmark110) |
| [about 133](#_bookmark130) |  | [about borderline elements 112](#_bookmark111) |
| [audited builds 134](#_bookmark131) |  | [checking results 113](#_bookmark112) |
| [case-sensitivity 50](#_bookmark53) |  | [error handling 113](#_bookmark112) |
| [configuring, and case-sensitivity 49](#_bookmark52) |  | [fixing symbolic links after 205](#_bookmark197) |
| [limitations on Windows 134](#_bookmark131) |  | [fixing view problems 204](#_bookmark196) |
| [locating data containers 207](#_bookmark199) |  | [precautions 112](#_bookmark111) |
| [recursive traversal of UNIX view root directory](#_bookmark202) | [210](#_bookmark202) | replicas |
| MVFS cache |  | [considerations for checkvob 179](#_bookmark174) |
| [and dynamic view performance 211](#_bookmark203) |  | [feature levels of 72](#_bookmark72) |
| [scaling factor 213](#_bookmark205) |  | [in administrative VOB hierarchies 106](#_bookmark106) |
| [statistics 212](#_bookmark204) |  | [moving 115](#_bookmark114) |
| [statistics available 221](#_bookmark211) |  | restoring data from backup |
| [mvfsstat command 221](#_bookmark211) |  | [about 155](#_bookmark151) |
|  |  | [administrative VOBs 167](#_bookmark163) |
|  |  | [ClearQuest databases 166](#_bookmark162) |
| **N** |  | [component VOBs 166](#_bookmark162) |

naming conventions

[file names, mixed case 49](#_bookmark52) [for VOBs 29](#_bookmark34)

NAS devices [about 3](#_bookmark8)

[and ClearCase 12](#_bookmark17)

[configuring network access to 13](#_bookmark18) [creating views on 138](#_bookmark135)

[creating VOBs on 78](#_bookmark78) [moving VOBs to 128](#_bookmark125)

[replacing VOB server host for 80](#_bookmark80)

[PVOBs 167](#_bookmark163)

[related databases 165](#_bookmark161)

[restoring individual elements 161](#_bookmark157) [synchronizing VOBs and views 163](#_bookmark159) [views 162](#_bookmark158)

[vob\_restore 155](#_bookmark151)

[VOBs on NAS devices 161](#_bookmark157)

[VOBs, procedure for standard 159](#_bookmark155) [rgy\_backup command 40](#_bookmark45)

[rgy\_switchover command 40](#_bookmark45)

[rmelem command 88](#_bookmark88)

[rmname command 88](#_bookmark88)

[root users, privileges of remote 56](#_bookmark58) [rwp process 10](#_bookmark15)

# S

[Samba, configuring for ClearCase 237](#_bookmark222) scheduler

[about 87,](#_bookmark87) [169](#_bookmark164)

[adjusting frequency of scrubbing jobs 178](#_bookmark173) [creating jobs 172](#_bookmark167)

[creating tasks 171](#_bookmark166)

[deleting jobs 175](#_bookmark170)

[deleting tasks 172](#_bookmark167)

[editing job properties 174](#_bookmark169) [editing tasks 171](#_bookmark166)

[job notifications 173](#_bookmark168)

[managing jobs 172](#_bookmark167)

[running jobs 175](#_bookmark170)

[scheduling jobs 172](#_bookmark167) [viewing job properties 174](#_bookmark169)

schema versions

[VOB database 72](#_bookmark72)

[scrubber utility 176,](#_bookmark171) [177](#_bookmark172) scrubbing

[about 88,](#_bookmark88) [176](#_bookmark171)

[storage pools 176](#_bookmark171)

[VOB databases 176](#_bookmark171) server storage locations

[about 31](#_bookmark36)

[creating 76,](#_bookmark76) [77](#_bookmark77)

[setuid execution, enabling in VOBs 82](#_bookmark82) site default values

[changing 37](#_bookmark42) snapshot views

[about 129,](#_bookmark126) [135](#_bookmark132)

[version selection 137](#_bookmark134)

[view database 136](#_bookmark133)

[view directory 135](#_bookmark132)

[view storage directory contents 136](#_bookmark133) source pools

[about 85](#_bookmark85)

[changing element assignment 91](#_bookmark91) [container protection on Windows 182](#_bookmark177) [data containers missing 183](#_bookmark178)

[patterns of use 90](#_bookmark90)

[unreferenced data container 183](#_bookmark178) storage directories

[access problems, troubleshooting 81](#_bookmark81) [access requirements for VOB 76](#_bookmark76) [creating Samba shares 239](#_bookmark224)

[dynamic view, contents and structure 130](#_bookmark127) [finding location of 148](#_bookmark144)

[fixing ACL problems 199](#_bookmark191) [preserving ACLs when copying 200](#_bookmark192)

[snapshot view, contents and structure 136](#_bookmark133) [VOB, content and structure 83](#_bookmark83)

storage management [adding pools 90](#_bookmark90)

[controlling VOB growth 176](#_bookmark171) [disk space for views 188](#_bookmark182) [techniques for VOBs 87,](#_bookmark87) [88](#_bookmark88)

storage pools [about 84](#_bookmark84)

[access control 62](#_bookmark64)

[additional, creating 90](#_bookmark90)

[fixing common problems 181](#_bookmark176)

[missing and unreferenced data containers 182](#_bookmark177)

storage pools *(continued)*

[moving to another disk 92](#_bookmark92)

[name and protection problems 181](#_bookmark176) [patterns of use 90](#_bookmark90)

[remote, accessed by Windows NFS client 235](#_bookmark220) [remote, and VOB backup 149](#_bookmark145)

[remote, consolidating 120](#_bookmark118)

[remote, finding 119](#_bookmark117) [remote, on UNIX hosts 90](#_bookmark90) [scrubbing 176](#_bookmark171)

[sources of inconsistencies 180](#_bookmark175) [stranded files, causes 89,](#_bookmark89) [131](#_bookmark128) [strip\_cr text mode 51](#_bookmark54)

# T

[TAS (TotalNET Advanced Server), configuring for](#_bookmark226) [ClearCase 241](#_bookmark226)

text modes

[about 50](#_bookmark53)

[criteria for choosing 52](#_bookmark55) [enabling support in replicas 53](#_bookmark56) [support in VOBs 52](#_bookmark55)

[types of 51](#_bookmark54)

The IBM Rational Web Platform (RWP) [about 275](#_bookmark253)

[transparent text mode 51](#_bookmark54) troubleshooting

[locating MVFS data containers 207](#_bookmark199) [protection on storage directories 199](#_bookmark191) [relocated elements and views 204](#_bookmark196)

[UNIX view root directory, recursive traversal by](#_bookmark202) [cronjob 210](#_bookmark202)

type objects

[*See also* global types](#_bookmark6) [about 94](#_bookmark94)

[converting instances to other types 96](#_bookmark96) [how defined 96](#_bookmark96)

[instances of 95](#_bookmark95)

[locking 63](#_bookmark65)

[scope 95](#_bookmark95)

[typographical conventions xv](#_bookmark2)

# U

UCM projects [importing data 191](#_bookmark184)

[using checkvob 188](#_bookmark182)

[UNIX kernel, resources on VOB host 75](#_bookmark75) UNIX/Windows interoperation

[about 45](#_bookmark48)

[characters not allowed in Windows file names 53](#_bookmark56) [cross-platform file-access solutions 231](#_bookmark216)

[text modes for views 50](#_bookmark53) [user credentials 45](#_bookmark48)

user administration [about 6](#_bookmark11)

[user names on UNIX and Windows 45](#_bookmark48)

# V

version selection

[in dynamic view 132](#_bookmark129) [in snapshot view 137](#_bookmark134)

[versions, removing 88](#_bookmark88)

view database

[contents for snapshot view 136](#_bookmark133) [view object 26](#_bookmark31)

[view root 130](#_bookmark127) view tags

[changing 36](#_bookmark41)

[contents and structure 28](#_bookmark33) [missing 39](#_bookmark44)

[removing 39](#_bookmark44)

[viewing contents 35](#_bookmark40) view\_server process

[about 8](#_bookmark13)

[view-private files, access control 66](#_bookmark68) views

[about 5,](#_bookmark10) [129](#_bookmark126)

[access control 64](#_bookmark66)

[backing up 151](#_bookmark147)

[cache size, how set 217](#_bookmark208) [cache size, resetting 219](#_bookmark210) [cache statistics 218](#_bookmark209)

[case-sensitive file lookup 50](#_bookmark53) [creating on NAS devices 138](#_bookmark135) [disabling temporarily 143](#_bookmark139)

[export, for non-ClearCase access 265](#_bookmark246) [how created 137](#_bookmark134)

[moving 138](#_bookmark135)

[registering 36](#_bookmark41)

[remote storage for 137](#_bookmark134) [removing 39,](#_bookmark44) [143](#_bookmark139) [restoring from backup 162](#_bookmark158) [starting 28](#_bookmark33)

[storage maintenance tasks 188](#_bookmark182) [synchronizing with VOBs 163](#_bookmark159) [text modes of 50,](#_bookmark53) [52](#_bookmark55)

[types of 129](#_bookmark126)

[Visual SourceSafe data, importing 191](#_bookmark184) VOB database

[contents and structure 86](#_bookmark86) [db\_server process 9](#_bookmark14)

[disk space requirements 75](#_bookmark75) [memory requirements 74](#_bookmark74)

[schema versions 72](#_bookmark72)

[scrubbing 176](#_bookmark171) VOB hosts

[about 3](#_bookmark8)

[configuring server processes 74](#_bookmark74)

[disk subsystem recommendations 225](#_bookmark212) [lock manager startup options 226](#_bookmark213) [performance tips 225](#_bookmark212)

[UNIX kernel, adjusting resources 75](#_bookmark75) [UNIX, remote storage pools 90](#_bookmark90)

[with many VOBs 226](#_bookmark213)

[with multiple network interfaces 14](#_bookmark19) VOB tags

[changing 36](#_bookmark41)

[content and structure 27](#_bookmark32) [missing 39](#_bookmark44)

[naming conventions 29](#_bookmark34)

[removing 39,](#_bookmark44) [83](#_bookmark83)

[viewing contents 35](#_bookmark40)

[vob\_restore utility 155,](#_bookmark151) [156,](#_bookmark152) [157](#_bookmark153)

[vob\_scrubber utility 176](#_bookmark171)

[vob\_server process 9](#_bookmark14)

[vob\_snapshot utility 147](#_bookmark143)

[vobrpc\_server process 9](#_bookmark14)