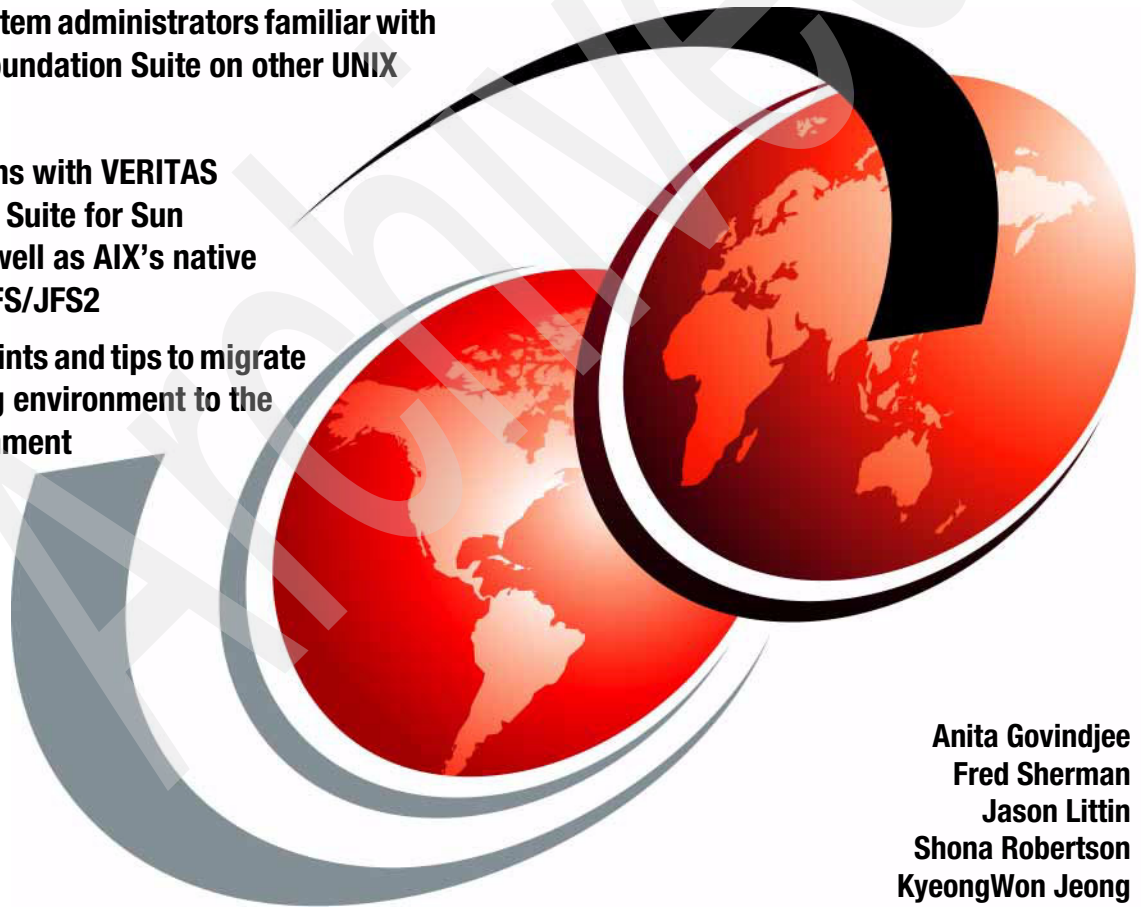


# Introducing VERITAS Foundation Suite for AIX

Helping system administrators familiar with  
VERITAS Foundation Suite on other UNIX  
platforms

Comparisons with VERITAS  
Foundation Suite for Sun  
Solaris as well as AIX's native  
LVM and JFS/JFS2

Providing hints and tips to migrate  
the existing environment to the  
AIX environment



Anita Govindjee  
Fred Sherman  
Jason Littin  
Shona Robertson  
KyeongWon Jeong





International Technical Support Organization

## **Introducing VERITAS Foundation Suite for AIX**

October 2002

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xiii.

### **First Edition (October 2002)**

This edition applies to IBM @server pSeries and RS/6000 Systems for use with the AIX 5L for POWER Version 5.1 Operating System, Program Number 5765-E61, and VERITAS Foundation Suite for AIX, and is based on information available in August 2002.

© Copyright International Business Machines Corporation 2002. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

|   |       |
|---|-------|
| <b>Figures</b> .....  | ix    |
| <b>Tables</b> .....   | xi    |
| <b>Notices</b> .....  | xiii  |
| Trademarks .....  | xiv   |
| <b>Preface</b> .....  | xvii  |
| The team that wrote this redbook .....                            | xviii |
| Become a published author .....                                   | xx    |
| Comments welcome .....  | xx    |
| <b>Chapter 1. Introduction</b> .....                              | 1     |
| 1.1 What is VERITAS Foundation Suite for AIX .....                | 2     |
| 1.1.1 Additional VERITAS products on AIX 5L Version 5.1 .....     | 3     |
| 1.1.2 Other supported platforms of VERITAS Foundation Suite ..... | 3     |
| 1.1.3 VERITAS NetBackup on AIX .....                              | 4     |
| 1.2 IBM-VERITAS relationship .....                                | 4     |
| 1.2.1 History of VERITAS .....                                    | 4     |
| 1.2.2 Joint agreements, offerings, and activities .....           | 4     |
| 1.2.3 VERITAS programs focused on interoperability .....          | 5     |
| 1.3 VERITAS Foundation Suite on the AIX Bonus Pack .....          | 6     |
| 1.4 Why use VERITAS Foundation Suite on AIX .....                 | 7     |
| 1.5 Support for LVM and JFS for AIX .....                         | 7     |
| <b>Chapter 2. Components</b> .....                                | 9     |
| 2.1 Overview .....  | 11    |
| 2.1.1 VERITAS Volume Manager overview .....                       | 11    |
| 2.1.2 VERITAS File System overview .....                          | 11    |
| 2.1.3 VERITAS Enterprise Administrator overview .....             | 11    |
| 2.2 VERITAS Volume Manager .....                                  | 12    |
| 2.2.1 Features .....  | 12    |
| 2.2.2 Physical objects .....                                      | 13    |
| 2.2.3 Device discovery .....                                      | 17    |
| 2.2.4 Enclosure-based naming .....                                | 18    |
| 2.2.5 Virtual objects .....                                       | 19    |
| 2.2.6 Volume layouts .....  | 24    |
| 2.2.7 Layered volumes .....                                       | 33    |
| 2.2.8 Online relayout .....                                       | 33    |

|  |           |
|--|-----------|
| 2.2.9 Hot relocation . . . . .                             | 33        |
| 2.2.10 Dirty Region Logging (DRL) . . . . .                | 33        |
| 2.3 VERITAS File System . . . . .                          | 34        |
| 2.3.1 Features . . . . .                                   | 34        |
| 2.3.2 Disk layout. . . . .                                 | 35        |
| 2.3.3 Extent-based allocation. . . . .                     | 36        |
| 2.3.4 Inodes . . . . .                                     | 37        |
| 2.3.5 Caching . . . . .                                    | 38        |
| 2.3.6 Journaling . . . . .                                 | 38        |
| 2.3.7 Online file system resizing . . . . .                | 39        |
| 2.3.8 Online defragmentation. . . . .                      | 39        |
| <b>Chapter 3. Planning and installation . . . . .</b>      | <b>41</b> |
| 3.1 Pre-installation planning . . . . .                    | 42        |
| 3.1.1 Hardware requirements. . . . .                       | 42        |
| 3.1.2 Operating system and software requirements . . . . . | 43        |
| 3.1.3 File system space . . . . .                          | 45        |
| 3.1.4 Licensing. . . . .                                   | 46        |
| 3.1.5 Selecting disks for use in VxVM . . . . .            | 46        |
| 3.2 Installation . . . . .                                 | 46        |
| 3.2.1 Installation using VRTSinstall . . . . .             | 48        |
| 3.2.2 Installation using SMIT . . . . .                    | 49        |
| 3.2.3 Installation using installp . . . . .                | 52        |
| 3.3 Post-installation tasks . . . . .                      | 53        |
| 3.3.1 Installing product licenses . . . . .                | 53        |
| 3.3.2 Initializing VERITAS Volume Manager . . . . .        | 54        |
| 3.3.3 Post-installation verification. . . . .              | 69        |
| 3.3.4 Uninstalling VxFS and VxVM . . . . .                 | 72        |
| <b>Chapter 4. Basic administration . . . . .</b>           | <b>77</b> |
| 4.1 System startup and process control . . . . .           | 79        |
| 4.1.1 Startup process . . . . .                            | 79        |
| 4.1.2 Managing processes . . . . .                         | 81        |
| 4.2 Methods of administration . . . . .                    | 83        |
| 4.2.1 Command-line interface . . . . .                     | 83        |
| 4.2.2 VEA Java GUI. . . . .                                | 84        |
| 4.2.3 VERITAS supplied utilities . . . . .                 | 87        |
| 4.2.4 Using SMIT . . . . .                                 | 87        |
| 4.3 Basic administration tasks. . . . .                    | 88        |
| 4.3.1 Adding disks . . . . .                               | 89        |
| 4.3.2 Creating disk groups . . . . .                       | 93        |
| 4.3.3 Creating volumes . . . . .                           | 98        |
| 4.3.4 Viewing VxVM object information . . . . .            | 112       |

|                   |  |            |
|-------------------|--|------------|
| 4.3.5             | Creating file systems . . . . .                      | 120        |
| 4.3.6             | Mounting file systems . . . . .                      | 122        |
| 4.3.7             | Resizing file systems . . . . .                      | 125        |
| 4.3.8             | Monitoring for failures . . . . .                    | 129        |
| <b>Chapter 5.</b> | <b>Advanced administration . . . . .</b>             | <b>131</b> |
| 5.1               | Dynamic multipathing . . . . .                       | 132        |
| 5.2               | Volume administration . . . . .                      | 136        |
| 5.2.1             | Monitoring tasks . . . . .                           | 136        |
| 5.2.2             | Creating volumes using vxmake . . . . .              | 137        |
| 5.2.3             | Adding a mirror to a volume . . . . .                | 140        |
| 5.2.4             | Removing a mirror from a volume . . . . .            | 142        |
| 5.2.5             | Adding a log to a volume . . . . .                   | 143        |
| 5.2.6             | Creating layered volumes . . . . .                   | 147        |
| 5.2.7             | Changing volume layouts . . . . .                    | 149        |
| 5.2.8             | Renaming volumes . . . . .                           | 150        |
| 5.2.9             | Removing volumes . . . . .                           | 151        |
| 5.3               | Disk group administration . . . . .                  | 152        |
| 5.3.1             | Adding and removing disks from disk groups . . . . . | 152        |
| 5.3.2             | Removing disk groups . . . . .                       | 153        |
| 5.3.3             | Deporting and importing disk groups . . . . .        | 153        |
| 5.4               | Backups and restores . . . . .                       | 156        |
| 5.4.1             | File system snapshots . . . . .                      | 156        |
| 5.4.2             | Volume snapshots . . . . .                           | 157        |
| 5.4.3             | Split mirror backups . . . . .                       | 160        |
| 5.4.4             | Using vxdump and vxrestore . . . . .                 | 163        |
| 5.5               | Problem prevention and resolution . . . . .          | 166        |
| 5.5.1             | Hot relocation . . . . .                             | 167        |
| 5.5.2             | Hot sparing . . . . .                                | 168        |
| 5.5.3             | Evacuating volumes from a disk . . . . .             | 169        |
| 5.5.4             | Replacing or removing disks . . . . .                | 171        |
| 5.6               | File system administration . . . . .                 | 175        |
| 5.6.1             | Setting block and intent log size . . . . .          | 175        |
| 5.6.2             | Quotas . . . . .                                     | 176        |
| 5.6.3             | Defragmenting file systems . . . . .                 | 178        |
| 5.6.4             | Optionally licensable features . . . . .             | 180        |
| <b>Chapter 6.</b> | <b>Comparisons . . . . .</b>                         | <b>181</b> |
| 6.1               | Comparisons with other UNIX platforms . . . . .      | 182        |
| 6.1.1             | ODM and SMIT integration . . . . .                   | 182        |
| 6.1.2             | Disk devices and the VxVM . . . . .                  | 184        |
| 6.1.3             | VxVM and LVM co-existence . . . . .                  | 185        |
| 6.1.4             | VxVM at system startup and shutdown . . . . .        | 186        |

|  |            |
|--|------------|
| 6.1.5 VxVM/VxFS command differences . . . . .                        | 187        |
| 6.1.6 VxVM/VxFS device drivers and kernel extensions . . . . .       | 187        |
| 6.1.7 Installation and packaging. . . . .                            | 189        |
| 6.1.8 The 64-bit kernel . . . . .                                    | 190        |
| 6.1.9 Debugging information . . . . .                                | 190        |
| 6.1.10 Dynamic MultiPathing (VxDMP) . . . . .                        | 192        |
| 6.2 AIX LVM, JFS/JFS2 and VxVM, VxFS compared . . . . .              | 193        |
| 6.2.1 Logical volume concepts. . . . .                               | 194        |
| 6.2.2 Volume layouts . . . . .                                       | 200        |
| 6.2.3 Backup . . . . .   | 206        |
| 6.2.4 Hot spare management. . . . .                                  | 208        |
| 6.2.5 JFS/JFS2 and VxFS comparison . . . . .                         | 208        |
| <b>Chapter 7. Migration considerations . . . . .</b>                 | <b>219</b> |
| 7.1 Reasons for migration . . . . .                                  | 220        |
| 7.2 Planning for migration . . . . .                                 | 221        |
| 7.2.1 Applications. . . . .  | 223        |
| 7.2.2 Operating system considerations . . . . .                      | 224        |
| 7.3 Migration of VxVM and VxFS on Solaris to AIX . . . . .           | 226        |
| 7.3.1 Test environment configuration. . . . .                        | 227        |
| 7.3.2 Cloning volume layouts . . . . .                               | 228        |
| 7.3.3 File system recreation . . . . .                               | 232        |
| 7.3.4 Tape backup and recovery . . . . .                             | 233        |
| 7.3.5 Using network facilities . . . . .                             | 234        |
| 7.3.6 Deport/import of VxVM disk groups . . . . .                    | 238        |
| 7.3.7 Using VERITAS Volume Replicator (VVR) . . . . .                | 238        |
| 7.4 Migration from AIX LVM to VxVM . . . . .                         | 239        |
| 7.4.1 Test environment configuration. . . . .                        | 239        |
| 7.4.2 Converting LVM volume groups to VxVM disk groups . . . . .     | 241        |
| 7.4.3 Manual migration of LVM and JFS to VxVM and VxFS . . . . .     | 251        |
| 7.5 Other migration scenarios . . . . .                              | 254        |
| 7.5.1 Single platform migration . . . . .                            | 254        |
| 7.5.2 Migration of a UNIX File system (UFS) to VxFS on AIX . . . . . | 255        |
| 7.6 Summary and recommendations . . . . .                            | 256        |
| <b>Chapter 8. Performance, tuning, and scalability . . . . .</b>     | <b>257</b> |
| 8.1 Basic performance guidelines for VxVM and VxFS. . . . .          | 258        |
| 8.1.1 Physical disks and data assignment . . . . .                   | 258        |
| 8.1.2 VxVM logs . . . . .  | 259        |
| 8.1.3 Extent-based allocation . . . . .                              | 263        |
| 8.1.4 Inode and directory optimizations . . . . .                    | 264        |
| 8.1.5 VxFS create options . . . . .                                  | 264        |
| 8.1.6 Mount command options . . . . .                                | 266        |



|  |            |
|--|------------|
| 8.2 Monitoring VxVM and VxFS . . . . .                                   | 268        |
| 8.2.1 vxstat and vxtrace . . . . .                                       | 268        |
| 8.3 Tuning . . . . .   | 271        |
| 8.3.1 VxVM global parameters . . . . .                                   | 271        |
| 8.3.2 VxFS global parameters . . . . .                                   | 272        |
| 8.3.3 Self tuning file systems . . . . .                                 | 272        |
| 8.3.4 File system tuning parameters . . . . .                            | 273        |
| 8.4 Application interface support . . . . .                              | 274        |
| 8.4.1 Cache advisories . . . . .   | 274        |
| 8.4.2 Other programatic advisories . . . . .                             | 275        |
| 8.5 Scalability . . . . .  | 275        |
| 8.5.1 Architectural scalability . . . . .                                | 276        |
| 8.5.2 Administrative scalability . . . . .                               | 277        |
| 8.5.3 Scaling services . . . . .   | 278        |
| <b>Chapter 9. Troubleshooting and technical support . . . . .</b>        | <b>279</b> |
| 9.1 How to get patches . . . . .   | 280        |
| 9.1.1 How to get patches from VERITAS . . . . .                          | 280        |
| 9.1.2 How to get patches from IBM . . . . .                              | 283        |
| 9.2 How to get technical support . . . . .                               | 285        |
| 9.2.1 How to get technical support from VERITAS . . . . .                | 285        |
| 9.2.2 How to get technical support from IBM . . . . .                    | 288        |
| 9.3 Installation issues . . . . .  | 290        |
| 9.3.1 VERITAS patches . . . . .  | 290        |
| 9.3.2 IBM APARs . . . . .  | 292        |
| 9.3.3 Possible installation issues . . . . .                             | 292        |
| 9.4 Administration issues . . . . .                                      | 295        |
| 9.5 References for troubleshooting . . . . .                             | 297        |
| <b>Appendix A. LVM and VxVM command comparison tables . . . . .</b>      | <b>299</b> |
| <b>Appendix B. JFS/JFS2 and VxFS command comparison tables . . . . .</b> | <b>305</b> |
| <b>Appendix C. Sample installation scripts . . . . .</b>                 | <b>309</b> |
| <b>Appendix D. The VERITAS Cluster Server for AIX . . . . .</b>          | <b>323</b> |
| Executive overview . . . . .   | 324        |
| Components of a VERITAS cluster . . . . .                                | 324        |
| Cluster resources . . . . .  | 325        |
| Cluster configurations . . . . .   | 328        |
| Cluster communication . . . . .  | 328        |
| Cluster installation and setup . . . . .                                 | 329        |
| Cluster administration facilities . . . . .                              | 330        |
| HACMP and VERITAS Cluster Server compared . . . . .                      | 330        |

|   |            |
|---|------------|
| Components of an HACMP cluster . . . . .                          | 331        |
| Cluster resources . . . . .                                       | 331        |
| Cluster configurations . . . . .                                  | 333        |
| Cluster communications . . . . .                                  | 333        |
| Cluster installation and setup . . . . .                          | 334        |
| Cluster administration facilities . . . . .                       | 335        |
| HACMP and VERITAS Cluster Server feature comparison summary . . . | 336        |
| <b>Abbreviations and acronyms . . . . .</b>                       | <b>339</b> |
| <b>Related publications . . . . .</b>                             | <b>341</b> |
| IBM Redbooks . . . . .  | 341        |
| Other resources . . . . .   | 341        |
| Referenced Web sites . . . . .                                    | 342        |
| How to get IBM Redbooks . . . . .                                 | 344        |
| IBM Redbooks collections . . . . .                                | 344        |
| <b>Index . . . . .</b>  | <b>345</b> |

# Figures

|      |  |     |
|------|--|-----|
| 2-1  | A disk array with three physical disks represented as one volume . . . . | 15  |
| 2-2  | A disk array with three physical disks represented as three volumes . .  | 16  |
| 2-3  | Disk enclosures and enclosure-based naming . . . . .                     | 18  |
| 2-4  | VM disk . . . . .  | 20  |
| 2-5  | Disk group containing three VM disks . . . . .                           | 20  |
| 2-6  | Subdisks . . . . .   | 21  |
| 2-7  | Plex with two subdisks . . . . .   | 22  |
| 2-8  | File system associated with volume. . . . .                              | 23  |
| 2-9  | Volume with one plex. . . . .  | 24  |
| 2-10 | Striping of a volume and its associated plex . . . . .                   | 26  |
| 2-11 | Mirroring a plex to plex. . . . .  | 27  |
| 2-12 | RAID 0+1 layout: striping with mirroring. . . . .                        | 28  |
| 2-13 | RAID 1+0 layout: mirroring with striping. . . . .                        | 30  |
| 2-14 | RAID 5 layout. . . . .   | 32  |
| 4-1  | VEA login screen . . . . .   | 85  |
| 4-2  | VEA main window . . . . .  | 86  |
| 4-3  | VEA add disk group screen . . . . .                                      | 94  |
| 4-4  | VEA create volume screen . . . . .                                       | 100 |
| 4-5  | VEA create file system window . . . . .                                  | 121 |
| 4-6  | VEA resize file system screen . . . . .                                  | 127 |
| 6-1  | LVM and VxVM logical and physical components . . . . .                   | 199 |
| 7-1  | Planning for migration of application and the operating system . . . .   | 222 |
| 7-2  | Planning for practical migration of data . . . . .                       | 223 |
| 7-3  | VxVM configuration options . . . . .                                     | 251 |
| 7-4  | VxFS configuration options . . . . .                                     | 252 |
| 9-1  | VERITAS Support Web site . . . . .                                       | 280 |
| 9-2  | VERITAS TechAlerts Product Selection page . . . . .                      | 281 |
| 9-3  | VERITAS File System for UNIX page containing AIX tech alert . . . .      | 282 |
| 9-4  | IBM eServer pSeries AIX 5L Technical Support Web site . . . . .          | 283 |
| 9-5  | Select AIX 5L fileset version to download . . . . .                      | 284 |
| 9-6  | Create new file system with VEA GUI . . . . .                            | 297 |

Archived

# Tables

|     |   |     |
|-----|---|-----|
| 3-1 | Minimum fileset levels required for installation              | 44  |
| 3-2 | Required file system space                                    | 45  |
| 3-3 | VxFS and VxVM packages  | 47  |
| 6-1 | Supported VxVM disk types Solaris and AIX                     | 185 |
| 6-2 | vxresize file system summary                                  | 185 |
| 6-3 | Supported software RAID levels for LVM/VxVM                   | 200 |
| 6-4 | Comparison of JFS2 and JFS features                           | 209 |
| 6-5 | JFS/JFS2 and VxFS comparison                                  | 211 |
| 6-6 | JFS and JFS2 inode structures                                 | 213 |
| 6-7 | VxFS inode structure  | 214 |
| 7-1 | User configuration files on IBM AIX compared with Sun Solaris | 225 |
| 8-1 | Volume group limits   | 276 |
| 8-2 | Physical volumes in volume/disk group                         | 277 |
| 8-3 | Mirror copies per logical volume                              | 277 |
| A-1 | Commands and tasks comparison in LVM and VxVM                 | 300 |
| A-2 | VxVM tasks with no LVM equivalent                             | 303 |
| A-3 | LVM tasks with no VxVM equivalent                             | 303 |
| B-1 | Commands and tasks comparison in JFS/JFS2 and VxFS            | 306 |
| B-2 | VxFS task with no JFS/JFS2 equivalent                         | 307 |
| B-3 | JFS/JFS2 task with no VxFS equivalent                         | 307 |
| D-1 | HACMP/VERITAS Cluster Server feature comparison               | 336 |
| D-2 | HACMP/VERITAS Cluster Server environment support              | 338 |



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:  
*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

|                            |   |          |
|----------------------------|---|----------|
| AIX®                       | IBM®  | Sequent® |
| AIX 5L™                    | Informix®   | SPTM     |
| BookMaster®                | MORE™   | SP2®     |
| DB2®                       | pSeries™  | Tivoli®  |
| DPI®                       | Redbooks™   | xSeries™ |
| Enterprise Storage Server™ | Redbooks(logo)™  |          |
| FlashCopy®                 | RS/6000®  |          |

The IBM eServer brand consists of the established IBM e-business logo with the following descriptive term 'server' following it.

The following terms are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both:

Notes®

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Itanium is a trademark of Intel Corporation.

VERITAS, the VERITAS Logo, VERITAS Bare Metal Restore, VERITAS Cluster Server, VERITAS Database Edition, VERITAS Enabled, VERITAS Enterprise Administrator, VERITAS File Replicator, VERITAS File System, VERITAS FlashSnap, VERITAS Foundation Suite, VERITAS Global Cluster Manager, VERITAS NetBackup, VERITAS QuickLog, VERITAS Volume Manager, VERITAS Volume Replicator and all other VERITAS product names and slogans are trademarks or registered trademarks of VERITAS Software Corporation. VERITAS and the VERITAS Logo Reg. U.S. Pat. & Tm. Off. Other product names and/or slogans mentioned herein may be trademarks or registered trademarks of their respective companies.

Sun, Sun Microsystems, the Sun logo, Sun Solve, Sun Cluster, Ultra, Solaris, Java, Java 3D, JVM, SunForum, SunVTS, SunFDDI, StarOffice, SunPCi, SunDocs, SunExpress, Open Windows, Solstice, Solstice AdminSuite, Solstice Backup, SPARCstorage, SunNet Manager, Online:DiskSuite, AutoClient, NFS, Solstice DiskSuite, Solaris Resource Manager, Solaris Bandwidth Manager, Solaris Web Start and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered



trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc.

Linux is a registered trademark of Linus Torvalds.

Other company, product, and service names may be trademarks or service marks of others.

Archived



# Preface

VERITAS Software products, such as VERITAS Volume Manager™ (VxVM) and VERITAS File System™ (VxFS), are popular on other UNIX platforms, and VERITAS announced these products for AIX in the early of May 2002. VERITAS Foundation Suite™ for AIX includes VERITAS Volume Manager for AIX and VERITAS File System for AIX. These products will be useful for people who are familiar with the same products on other UNIX platforms since VERITAS Foundation Suite for AIX provides a similar environment for them. That means they can use their same skills and knowledge in the AIX environment.

This IBM Redbook will not only help system administrators who are familiar with VERITAS Foundation Suite on other UNIX platforms but also AIX system administrators who are familiar with AIX's native Logical Volume Manager (LVM) and Journaled File Systems (JFS) and would like to learn more about VERITAS Foundation Suite for AIX.

This redbook will compare VERITAS Foundation Suite for AIX with VERITAS Foundation Suite for Sun Solaris as well as AIX's native LVM and JFS/JFS2. One of the main focuses is the procedures, hints, and tips to migrate the existing environment to the AIX environment. It will also cover all the details about the functionalities of these products, including planning, installation, configuration, administration, performance, tuning, and troubleshooting.

This redbook also introduces VERITAS Cluster Server™ in Appendix D, "The VERITAS Cluster Server for AIX" on page 323 for those interested in setting up a clustered environment.

This redbook will cover the following topics:

- ▶ Chapter 1 - Introduction. In this chapter, we introduce VERITAS Foundation Suite for AIX for the IBM AIX 5L Version 5.1 operating system.
- ▶ Chapter 2 - Components. In this chapter, we introduce the terminology and concepts used in VERITAS Foundation Suite. This is a good chapter to start with before plunging into the installation and administration of your system.
- ▶ Chapter 3 - Planning and installation. We discuss pre-installation issues of your data management system and how to do an installation for different configurations in this chapter.
- ▶ Chapter 4 - Basic administration. This chapter goes over the steps in administering a basic VERITAS Foundation Suite setup.

- ▶ Chapter 5 - Advanced administration. Here we discuss some more advanced topics in administering your system using VERITAS Foundation Suite for AIX.
- ▶ Chapter 6 - Comparisons. In this chapter, we show how the concepts and terminology used with VERITAS Foundation Suite for AIX compares to that used with the native AIX offerings of LVM, JFS, and JFS2. We also discuss the coexistence of LVM with VxVM, as well as the coexistence of IBM storage subsystems with VERITAS Foundation Suite for AIX.
- ▶ Chapter 7 - Migration considerations. Here we discuss how you would accomplish various types of data migrations. Included in this chapter are descriptions on how to migrate your data from a SUN Solaris system to AIX, as well as other migration scenarios.
- ▶ Chapter 8 - Performance, tuning, and scalability. This chapter considers what influences performance of VERITAS Foundation Suite for AIX software. Scalability is also discussed.
- ▶ Chapter 9 - Troubleshooting and technical support. This chapter contains information regarding how to obtain technical support from both VERITAS and IBM, as well as how to use the VRTSexplorer. It also describes possible technical issues that you may encounter during installation and administration of VERITAS Foundation Suite for AIX.
- ▶ Appendix A - LVM and VxVM command comparison tables. These are tables comparing common volume management tasks and commands.
- ▶ Appendix B - JFS/JFS2 and VxFS command comparison tables. These are tables comparing common file system management tasks and commands.
- ▶ Appendix C - Sample installation scripts. This appendix provides scripts that you can use on your AIX systems to install and configure VERITAS Foundation Suite for AIX on multiple systems.
- ▶ Appendix D - VERITAS Cluster Server for AIX. This appendix introduces VERITAS Cluster Server for AIX.

## The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

**KyeongWon Jeong** is a Consulting I/T Specialist at the International Technical Support Organization, Austin Center. He writes extensively on AIX and education materials and teaches IBM classes worldwide on all areas of AIX. Before joining the ITSO three and half years ago, he worked in IBM Global Learning Services in Korea as a Senior Education Specialist and was a class manager of all AIX classes for customers and interns. He has many years of teaching and

development experience. He is an IBM Certified Advanced Technical Expert - RS/6000 AIX.

**Anita Govindjee** is a technical consultant with IBM in the United States. She works in the IBM Server Group's pSeries Solutions Development team doing ISV technical support. She has more than ten years of experience doing C/C++ and Java software development, and AIX, Sun Solaris, and HP-UX system administration. She has spent the past two years providing technical support to ISVs on pSeries and AIX issues. She holds a BS from the University of Illinois, Urbana-Champaign, and a MS degree from Stanford University, both in Computer Science. She has written several whitepapers, including a Solaris to AIX migration whitepaper and an article on using the GNU C/C++ compiler on AIX.

**Fred Sherman** is a Principal Systems Engineer with the IBM Strategic Alliances Group, VERITAS Software Corporation. He has over 13 years of systems engineering experience on a variety of hardware and operating systems. His areas of expertise include high availability, performance and tuning, and systems architecture. He has certifications for AIX, SP2, Solaris and Oracle.

**Jason Littin** is a UNIX system administrator in Auckland, New Zealand. He has five years of experience in UNIX system administration on Sun's Solaris, IBM AIX and HP's HP/UX operating systems. His areas of expertise include administration of products such as VERITAS Netbackup, Tivoli Storage Manager, and VERITAS Foundation Suite on Solaris platforms.

**Shona Robertson** is a pre-sales specialist working in the advanced technical support team in the UK. She has worked with RS/6000 and pSeries since joining IBM five years ago and is also a member of the UK High Availability Cluster Competence Centre. Previously, she was an HACMP and Centre Technical Advisor in the UK UNIX Support Centre.

Thanks to the following people for their contributions to this project:

Wade Wallace  
International Technical Support Organization, Austin Center

Ella Buslovich  
International Technical Support Organization, Poughkeepsie Center

Johnny Shieh and Roberto Nava  
IBM Austin

Lawrence Wilson  
IBM Poughkeepsie

John Easton  
IBM U.K.

Deniz Erguvan  
IBM Germany

Ram Pandiri, Anindya Mukherjee, Paul Massiglia, Narasimha Valiveti, Amit Bothra, Dan Liport, Jade Arrington, Sathish Nayak, Dhiren Patel, Umesh Toprani, Ari Plaut, Masoud Sadrolashrafi, Patrick Carri, Poorna Ambati  
VERITAS Software Corporation

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an Internet note to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. JN9B Building 003 Internal Zip 2834

11400 Burnet Road  
Austin, Texas 78758-3493

Archived





# Introduction

This chapter introduces VERITAS Foundation Suite for AIX for the IBM AIX 5L Version 5.1 operating system.

This chapter discusses the following topics:

- ▶ What is VERITAS Foundation Suite for AIX
- ▶ IBM-VERITAS relationship
- ▶ VERITAS Foundation Suite on the AIX Bonus Pack
- ▶ Why use VERITAS Foundation Suite on AIX
- ▶ Support for LVM and JFS for AIX

## 1.1 What is VERITAS Foundation Suite for AIX

VERITAS Foundation Suite for AIX has recently been announced for the IBM AIX 5L Version 5.1 operating system. VERITAS NetBackup has been available for some years on IBM's AIX platform, but beginning in May 2002, VERITAS Foundation Suite is available.

VERITAS Foundation Suite for AIX is comprised of two base products: VERITAS Volume Manager (VxVM) and VERITAS File System (VxFS), plus VERITAS Enterprise Administrator (VEA) Graphical User Interface (GUI). VVR and VCS are separate products that require separate licenses. VERITAS FlashSnap is an advanced feature of VERITAS Foundation Suite for AIX that requires a separate license key. Note that VxVM and VxFS are not available as separate products on the AIX 5L Version 5.1 platform.

VERITAS Volume Manager is a simple to use, yet powerful disk and storage management system for enterprise computing. It supports online disk management, thus affording continuous data availability. Disk configuration can be done online without impacting users. VxVM also supports disk striping and disk mirroring. For data redundancy and protection against disk and hardware failures, VxVM supports RAID levels RAID 0 (disk striping), RAID 1 (disk mirroring), RAID 5, RAID 0+1, and RAID 1+0.

VERITAS File System is a reliable, scalable, fast-recovery journaling file system with increased data availability and data integrity features. Data availability is at the level necessary for mission-critical systems, where file system data is available within seconds of a system crash and reboot. Data integrity is maintained through the journaling file system that records changes in an intent log and then recovers from a crash using that log. Online management features are available with VxFS, such as file system backup, defragmentation, and growing and shrinking file systems.

The VERITAS Enterprise Administrator (VEA) GUI is provided with VERITAS Foundation Suite for AIX and supports both VxVM and VxFS. VEA enables easy online volume management and file system management. This is available not only for managing a set of AIX machines, but in a heterogeneous environment with many platforms, VEA can be used to do disk management across all the platforms simultaneously. From just one VEA console, multiple hosts and operating systems can be managed.

### 1.1.1 Additional VERITAS products on AIX 5L Version 5.1

The following VERITAS products are also available on AIX 5L Version 5.1:

- ▶ VERITAS Foundation Suite/HA for AIX (includes VERITAS Foundation Suite and VERITAS Cluster Server)
- ▶ VERITAS Volume Replicator for AIX (VVR)
- ▶ VERITAS Cluster Server for AIX (VCS)
- ▶ VERITAS FlashSnap for AIX (a separately licensed component of VERITAS Foundation Suite)
- ▶ VERITAS Database Edition for DB2 on AIX
- ▶ VERITAS Database Edition/HA for DB2 on AIX (includes VCS and VERITAS Cluster Server Agent for DB2)
- ▶ VERITAS Database Edition for Oracle on AIX
- ▶ VERITAS Database Edition/HA for Oracle on AIX (includes VCS and VERITAS Cluster Server Agent for Oracle)

The agents listed below are available as options for VERITAS Cluster Server for AIX and for VERITAS NetBackup for AIX. These are separately licensed components of VERITAS Cluster Server and VERITAS NetBackup, respectively:

- ▶ VERITAS Cluster Server Agent for DB2 on AIX
- ▶ VERITAS Cluster Server Agent for Oracle on AIX
- ▶ VERITAS NetBackup Agent for DB2 on AIX
- ▶ VERITAS NetBackup Agent for Oracle on AIX

Other VERITAS NetBackup agents are also available for Sybase, Informix, and Notes.

### 1.1.2 Other supported platforms of VERITAS Foundation Suite

VERITAS Foundation Suite is available on the following platforms:

- ▶ IBM AIX 5L Version 5.1.
- ▶ HP-UX 11.0 and 11i.
- ▶ Sun Solaris 2.6, 7, 8, and 9.
- ▶ Red Hat Linux 7.1 errata and 7.2.
- ▶ VERITAS Volume Manager for Windows 2000 and Windows NT are also available.

The components of VERITAS Foundation Suite: VERITAS Volume Manager and VERITAS File System, have been available since 1991.

### **1.1.3 VERITAS NetBackup on AIX**

VERITAS NetBackup has been available on AIX since AIX Version 4.1. The full suite of VERITAS software products now joins VERITAS NetBackup on AIX.

## **1.2 IBM-VERITAS relationship**

IBM and VERITAS have forged a strong relationship that includes marketing and development and support agreements, as well as educational offerings and solutions available from VERITAS Services and IBM Global Services. IBM and VERITAS are business partners working closely together to provide VERITAS products, services, and support.

### **1.2.1 History of VERITAS**

VERITAS was founded in 1989 and is a leading provider of storage management software for data protection, data availability, and disaster recovery. VERITAS is one of the fastest growing companies and has grown steadily since its inception, including successfully merging and acquiring several companies. VERITAS leads in the marketplace with 86 percent of Fortune 500 companies relying on VERITAS storage management solutions, across diverse server platforms and applications, and across different storage hardware and appliances.

### **1.2.2 Joint agreements, offerings, and activities**

IBM and VERITAS have a joint development and marketing agreement in place to deliver VERITAS products and services. VERITAS products and services for the IBM AIX platform are available directly from VERITAS. IBM Global Services also resells all VERITAS products on all supported platforms, including IBM AIX 5L Version 5.1 operating system. Additionally, an evaluation version of VERITAS Foundation Suite for AIX is available on the July 2002 Bonus Pack for IBM AIX 5L Version 5.1 (see 1.3, “VERITAS Foundation Suite on the AIX Bonus Pack” on page 6 for more information).

The joint development agreement between IBM and VERITAS includes development not only on AIX 5L Version 5.1 but also on the Windows platform and on Linux on the IBM @server xSeries platform.

At IBM, all VERITAS storage management solutions are also available from IBM Global Services, which has a practice devoted solely to VERITAS products and services.

IBM's and VERITAS Software's relationship includes integrated product development for relevant products. VERITAS has early access to AIX releases,

thereby keeping current with the newest AIX versions. VERITAS also works closely with the AIX kernel team. In fact, VERITAS is the first vendor that has had kernel modifications made to the AIX kernel for its software. Additionally, new versions of VERITAS products will stay current on AIX, just as on other VERITAS-supported UNIX platforms, such as SUN Solaris or HP-UX. Just as with software development agreements, relevant hardware, such as new IBM disk arrays and tapes, are also part of the development agreement. New IBM hardware will be qualified for use with VERITAS products.

A cooperative support agreement is in place, and IBM and VERITAS technical support teams have cross-trained on each others' products to provide the best service for the customer. A technical support call into either VERITAS or IBM regarding VERITAS software on IBM platforms will result in a warm transfer between the two companies at the appropriate support level. This support is available worldwide.

VERITAS Education Services provide courses that give training on VERITAS Foundation Suite and other VERITAS products. These courses address using VERITAS on the AIX 5L Version 5.1 operating system, as well as other operating systems. IBM eServer pSeries machines are used in some courses. Types of training include direct and indirect channel training, and customer training. A new AIX Field Technical Specialists organization is in place at VERITAS to provide field-based expertise. This organization is part of the Worldwide Field Specialists organization and is mandated to help both IBM and VERITAS with pre-sales information.

Dedicated alliance teams are at both IBM and VERITAS to facilitate communication between the two companies. These strategic alliance teams are at many levels, including business development and the sales organizations for both companies.

The marketing and development activities at the two companies involve sharing requirements, jointly identifying market opportunities, and joint testing during development, as well as joint standards collaboration.

### **1.2.3 VERITAS programs focused on interoperability**

This section describes VERITAS Software's two programs, which are VERITAS Enabled and VERITAS Powered, focused on interoperability.

#### **VERITAS Enabled**

The VERITAS Enabled program expands and enhances VERITAS Software's long history of partner collaboration to create a rich, flexible, and scalable framework for cooperative activities. With VERITAS Enabled, hardware and software vendors are able to integrate their solutions with VERITAS Software's

storage infrastructure, high availability, and data protection software solutions, supporting the industry's widest selection of applications, operating systems, servers, storage arrays, and appliances.

Hardware vendors who support the VERITAS Enabled program are able to leverage VERITAS software storage device plug-in layer, enabling VERITAS and third-party software vendors to take advantage of such array and fabric capabilities as multipathing, LUN configuration, snapshot, replication, virtualization mapping, LUN zoning, LUN masking, and discovery.

Software vendors who support the VERITAS Enabled program utilize an application plug-in layer that enables access to management interfaces for many of VERITAS Software's products, and access to agents that extend high availability and data protection solutions to specific application and database environments. VERITAS Software provides unique storage management capabilities for databases and enterprise applications.

### **VERITAS Powered**

VERITAS Powered is a strategic initiative to extend VERITAS Software's storage intelligence to new networking hardware platforms. Launched with broad support from established and emerging hardware partners, VERITAS Powered will create a wide range of next-generation storage networking products that address customer demand for efficient, reliable, and centrally managed storage.

New storage networking technologies allow storage software intelligence to reside outside traditional hardware servers and disk arrays and within a wide variety of devices, including switches, routers, appliances, and network-attached storage (NAS) subsystems. Together with its VERITAS Powered partners, VERITAS Software is optimizing its proven server-based data protection, network-attached storage, and storage infrastructure products for these emerging platforms.

The new storage networking solutions will give companies the flexibility to migrate to new topologies and hardware platforms, while continuing to benefit from the interoperable and trusted storage software they use today from VERITAS Software. VERITAS Powered solutions will be integrated with traditional VERITAS server-based software to enable a consistent, seamless customer experience. This ensures companies can simplify their storage management processes while embracing new innovations in storage technology.

## **1.3 VERITAS Foundation Suite on the AIX Bonus Pack**

An evaluation version of VERITAS Foundation Suite for AIX and Foundation Suite/HA for AIX are both available on the AIX 5L Version 5.1 July 2002 Bonus

Pack. The Foundation Suite/HA is the high availability version of the Foundation Suite, and includes VERITAS Cluster Server. Both VERITAS Foundation Suite for AIX and Foundation Suite/HA for AIX are full-featured versions of the software. Once you have installed the software, you need to request a demo license directly from VERITAS. The demo license is valid for 60 days.

## 1.4 Why use VERITAS Foundation Suite on AIX

Although the IBM AIX operating system has its own native Logical Volume Manager (LVM) and Journaled File System (JFS) that provide similar functionality as the VERITAS Foundation Suite components, there are compelling business reasons to use VERITAS Foundation Suite for AIX. The key differentiator is the common cross platform management and integration.

For organizations that already have the required skill base on VERITAS Foundation Suite on other platforms, such as SUN Solaris, HP-UX, or others, there is an easy migration from those platforms to AIX. No additional storage management software training is required to support the AIX platform. The functionality of VERITAS Foundation Suite on other supported platforms is the same as that on AIX. The GUI interface provided with VERITAS Foundation Suite is common across Solaris, AIX 5L, Windows, HP-UX, and Linux. Additionally, users can take advantage of the comprehensive features of VERITAS Foundation Suite for AIX, which are described in the following chapters.

One of the most important reasons for using VERITAS Foundation Suite on AIX is the ease of use in a heterogeneous environment with servers from IBM, SUN, HP, and others. In a heterogeneous environment, being able to use one common storage management software system makes the administrator's job much simpler. Common storage management lowers overall administrative costs and gives better total cost of ownership by reduced training costs. By using VERITAS Foundation Suite on AIX, the power of VERITAS software is available on the wide range of IBM @server pSeries servers, providing world-class solutions for organizations.

## 1.5 Support for LVM and JFS for AIX

In this redbook, we discuss using VERITAS Foundation Suite for AIX. We would like to note, though, that IBM continues to support the native Logical Volume Manager (LVM) and Journaled File System (JFS) for IBM AIX 5L Version 5.1. LVM and JFS are strategic products for IBM, and continue to be developed and enhanced. In fact, with AIX 5L Version 5.1, an enhanced and updated version of

the JFS is available called the Enhanced Journaled File System (JFS2). JFS2 has new additional features that include extent-based allocation, B-tree sorted directories for faster directory access, dramatically increased maximum file system size (4 PB in JFS2 versus 1 TB in JFS), increased maximum file size, and dynamic space allocation for file system objects. LVM, JFS, and JFS2 are built in to the AIX base operating system and are provided with any base AIX installation.

It is possible for LVM and JFS/JFS2 to easily co-exist with VERITAS Foundation Suite on the same AIX machine. It is possible to have the LVM and JFS/JFS2 used for one physical volume and VERITAS Volume Manager and VERITAS File System used on another physical volume on the same machine.



# Components

This chapter describes the underlying components of VERITAS Foundation Suite for AIX, for both VERITAS Volume Manager and for VERITAS File System. We also describe the concepts and terminology used by VERITAS Foundation Suite for AIX.

In this chapter, the following topics are described:

- ▶ Overview
  - VERITAS Volume Manager overview
  - VERITAS File System overview
  - VERITAS Enterprise Administrator overview
- ▶ VERITAS Volume Manager
  - Features
  - Physical objects
  - Device discovery
  - Enclosure-based naming
  - Virtual objects
  - Volume layouts
- ▶ VERITAS File System
  - Features

- Disk layout
- Extent-based allocation
- Journaling
- Inodes
- Caching
- Online file system resizing
- Online defragmentation

## 2.1 Overview

VERITAS Foundation Suite for AIX is comprised of two base products: VERITAS Volume Manager (VxVM) and VERITAS File System (VxFS), along with VERITAS Enterprise Administrator (VEA) graphical user interface (GUI).

### 2.1.1 VERITAS Volume Manager overview

Let us first look at VERITAS Volume Manager. Without a logical volume manager, such as VxVM, the system administrator of a machine is constrained by the physical limitations of the disk devices. By using VxVM, we overcome the physical limitations of the disk devices by combining the physical storage into virtual volumes. The virtual volumes behave just like physical devices to the applications using the volumes. By removing the restrictions of allocating disk space within a particular physical volume, these virtual volumes can provide greater capacity, availability, and performance than any given physical device. Concatenation, Redundant Array of Independent Disks (RAID) layouts, such as striping, mirroring, and RAID 5, are all supported by VxVM. VxVM also supports hot spares, thus improving data availability.

In this chapter, we describe the components, concepts, and terminology used by VxVM in creating virtual volumes from physical volumes.

### 2.1.2 VERITAS File System overview

The file system is a core element of any UNIX system, including AIX. The file system is what applications use to access data on the physical disk devices. VERITAS File System (VxFS) works hand in hand with VxVM. VxVM is used to create and manage volumes, and VxFS is used to manage the file systems that are formatted on these volumes.

VERITAS File System provides for online system backup, journaling for data integrity, online defragmentation, online file system resizing, and file system management in a heterogeneous environment. These and other features of VERITAS File System are described in 2.3, “VERITAS File System” on page 34.

### 2.1.3 VERITAS Enterprise Administrator overview

VERITAS Foundation Suite for AIX includes the VERITAS Enterprise Administrator (VEA) GUI. VEA can be used for both volume management and for file system management. The VEA GUI is Java-based, and is the same across platforms. In a heterogeneous environment, across different platforms, it is useful to have a single tool such as VEA to do file and volume management.

## 2.2 VERITAS Volume Manager

VxVM is a simple-to-use online disk storage management software system. VxVM aggregates or subdivides the storage capacity of disk drives (or RAID system LUNs) to increase capacity, I/O performance, or availability, or combinations of the three. The VxVM functionality can be accessed through the VEA GUI or through command-line interface (CLI) commands. VxVM supports RAID, providing data protection and data availability. Supported RAID levels are RAID 0 (Striping), RAID 1 (Mirroring), RAID 0+1 (Striping with Mirroring), RAID 1+0 (Mirroring with Striping), and RAID 5 (Striping with Parity). These RAID levels are described in 2.2.6, “Volume layouts” on page 24.

VxVM can coexist with AIX’s native LVM. You can decide which volumes you want managed by each volume manager. The **vxvmconvert** utility, included with VxVM, can be used to migrate LVM volume groups to VxVM disk groups. See 7.4.2, “Converting LVM volume groups to VxVM disk groups” on page 241 for more information. VERITAS Volume Manager only manages non-rootvg (non-root volume group) disks at this time.

### 2.2.1 Features

VERITAS Volume Manager offers the features shown in this section.

#### High availability

- ▶ Online storage reconfiguration: You can do storage management tasks online.
- ▶ Mirroring: See “Mirroring (RAID 1)” on page 26.
- ▶ Data Redundancy (RAID 1, RAID 0+1, RAID 1+0, RAID 5): See “Mirroring (RAID 1)” on page 26, “Striping with mirroring (RAID 0+1)” on page 28, “Mirroring with striping (RAID 1+0)” on page 29, “Striping with parity (RAID 5)” on page 30.
- ▶ Hot relocation of failed redundant storage: See 2.2.9, “Hot relocation” on page 33.
- ▶ Unrelocate.
- ▶ Dirty Region Logging (DRL): See 2.2.10, “Dirty Region Logging (DRL)” on page 33.

#### High performance and scalability

- ▶ Online performance monitoring and tuning tools
- ▶ Striping: See “Striping (RAID 0)” on page 25

- ▶ Dynamic multipathing (DMP) support: See “Dynamic Multipathing (DMP)” on page 16
- ▶ Spanning of multiple disks
- ▶ Task Monitor
- ▶ VERITAS FlashSnap (separately licensed option)

### **Centralized management**

- ▶ Dynamic Reconfiguration support
- ▶ Intuitive graphical user interface (VEA): See 2.1.3, “VERITAS Enterprise Administrator overview” on page 11
- ▶ Management of free-space pool for volume growth
- ▶ Maximum volume size button
- ▶ Unique disk IDs
- ▶ Logical disk groups: See “Disk groups” on page 20
- ▶ Multiple disk spanning

### **Support across multiple hardware and platforms**

- ▶ Ability to move disk groups between different supported disks
- ▶ Support of multiple disk arrays
- ▶ Heterogeneous platform availability
- ▶ Platform-independent graphical user interface

### **Integration**

- ▶ Highly integrated with other VERITAS storage applications

## **2.2.2 Physical objects**

VxVM operates on physical objects that are represented by virtual objects. Physical objects are physical hardware disks that house the actual data.

Virtual objects, or volumes, are linked to physical disks. Each volume can manipulate data on one or more physical disks. The application layer interacts with the volumes as if the application were accessing the physical disks directly. There are other virtual objects that we will discuss in the next section which are sub- or super-elements of volumes, such as subdisks, plexes, disk groups, or VM disks.

## Physical disks

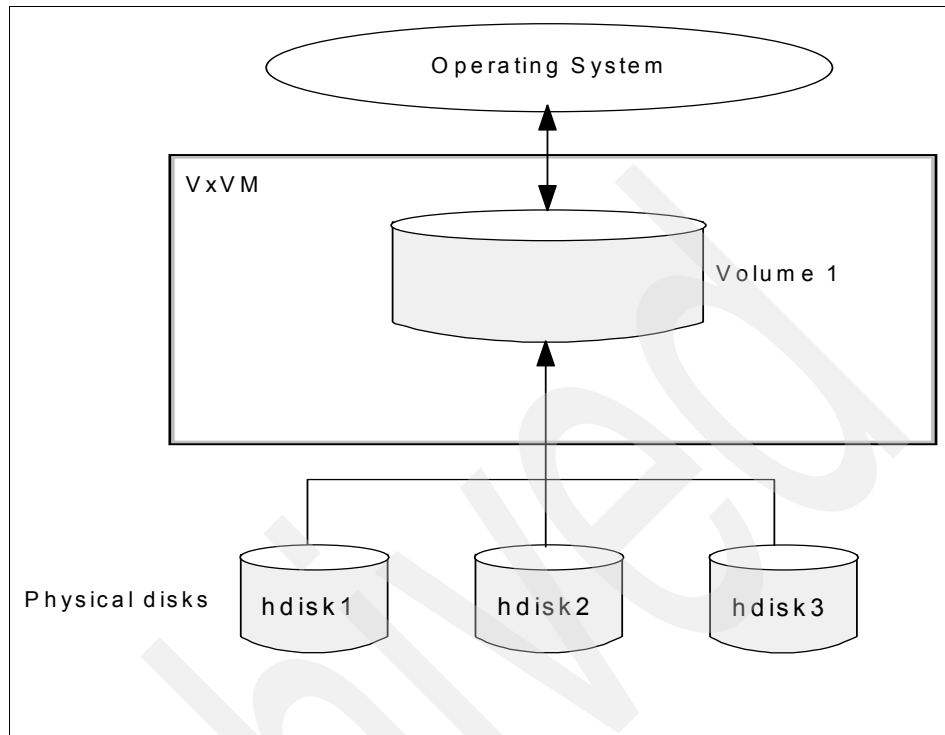
A physical disk is the basic hardware device where the data actually resides. The physical device can be accessed by name. For AIX 5L Version 5.1, typical device names are of the form `hdisk##`, for example, `hdisk1`.

When VxVM creates a virtual object to identify a given physical disk, VxVM can then identify a physical disk even if there is a system outage or other failure. This provides for quick failure detection and data recovery.

## Disk arrays

VxVM can treat a set of physical disks as a single entity, which makes it easier to provide data protection and data availability. For example, if we want to protect the data that resides on several physical disks, we could do so via mirroring (duplicating) the data onto another set of physical disks. VxVM can treat the set of physical disks where the data resides as a single atomic unit, and mirror that atomic unit to another set of disks. This approach makes the disk management easier.

A collection of physical disks is called a *disk array*. This can be represented as one or more virtual objects, or volumes. As stated above, the VxVM volumes can be manipulated by the operating system just the same as physical disks. At the application layer, applications also work with the volumes just as if they were physical disks. In Figure 2-1 on page 15, we show three physical disks, `hdisk1`, `hdisk2`, and `hdisk3`, which are represented as one volume.



*Figure 2-1 A disk array with three physical disks represented as one volume*

And in Figure 2-2 on page 16, we show another disk array where the same three physical disks, hdisk1, hdisk2, and hdisk3, are represented as three volumes.

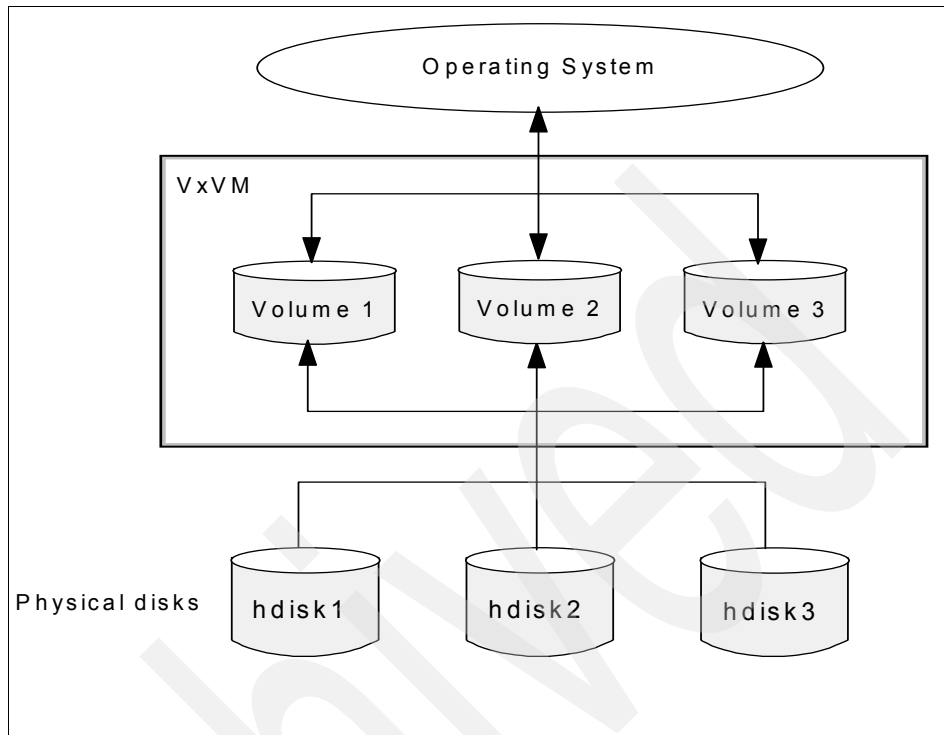


Figure 2-2 A disk array with three physical disks represented as three volumes

### Multipath disk arrays

Some disk arrays provide multiple I/O ports to access the disk devices. These ports provide multiple hardware paths for read/write access to the disk devices. Such disk arrays are called multipath disk arrays. By having multiple hardware paths to access the same disk device, data can be routed through an alternate pathway if one pathway fails.

### Dynamic Multipathing (DMP)

Dynamic Multipathing (DMP) provides the software support of a multipath disk device. DMP has two main advantages:

- ▶ Improved performance through load balancing
- ▶ Improved data availability through path failover

For load balancing, let us look at the following example. We have two paths to a particular disk device. Using DMP, VxVM can transfer data across both paths, thus distributing the data over the two paths. This decreases the amount of I/O going across each one of the paths. We therefore achieve better I/O performance



and throughput through balancing the I/O across the two pathways. The other main advantage of DMP is using the multiple paths for path failover. Path failover automatically and transparently redirects requests from the failed path to an alternate path, providing for greater data availability. For example, if we again have a disk device that has two paths to it, and one path fails, the data can still be routed through the remaining path.

Metanodes are used to access the disks in a disk array. One metanode is assigned to the set of paths connected to a particular disk device. DMP also uses the metanode to associate the multipathing policy that is appropriate for that disk array.

The type of multipathing used by DMP depends on the type of disk array. Active/active disk arrays primarily provide for load balancing and active/passive disk arrays primarily provide for path failover.

### ***Active/active disk arrays***

Active/active disk arrays allow I/O to go through multiple paths simultaneously. These types of disk arrays provide greater I/O throughput by doing load balancing of the I/O across the multiple paths to the disk devices. DMP in VxVM uses a *balanced path* policy to distribute I/O across the available paths to the active/active disk arrays. The method used to balance the I/O does the following:

- ▶ Sequential I/O starting within a certain range goes through the same path to optimize I/O throughput.
- ▶ I/O that does not fall within this particular range is distributed across multiple paths to take advantage of load balancing.

In other words, the I/O load is balanced across all the available paths, and sequential I/O within a particular range is sent down the same path.

### ***Active/passive disk arrays***

Active/passive disk arrays use a primary-secondary scheme. There is only one primary path that is used for I/O at a time. In the event of a failure, then a secondary path is used for I/O. This is a method that enhances data availability through failover. Active/passive disk arrays guarantee that if there is a failure of the primary or secondary path, data will continue to be available.

## **2.2.3 Device discovery**

Device discovery is exactly what it sounds like: discovering the disk devices attached to a system. Device discovery allows you to dynamically add new disk arrays and the corresponding DMP configuration of that array without having to do a reboot. The new disk array is dynamically added to the disk device database.

## 2.2.4 Enclosure-based naming

In “Physical disks” on page 14, we described a default method of naming physical disks and showed an example of `hdisk1`. There is another method of naming disk devices called enclosure-based naming. Disk naming can be arbitrary, and it can be useful to name disks according to the enclosures in which they are mounted. Enclosure-based naming can be used to refer to each disk within an enclosure.

In a typical Storage Area Network (SAN) environment, host controllers can be connected to multiple enclosures through a Fibre Channel hub or fabric switch. In a SAN, information about disk location provided by the operating system may not correctly indicate the physical location of the disks.

In Figure 2-3, we see the device names for the disks in enclosure `enc0` are named `enc0_0`, `enc0_1`, and so on. The main benefit of this scheme is that it allows you to quickly determine where a disk is physically located in a large SAN configuration.

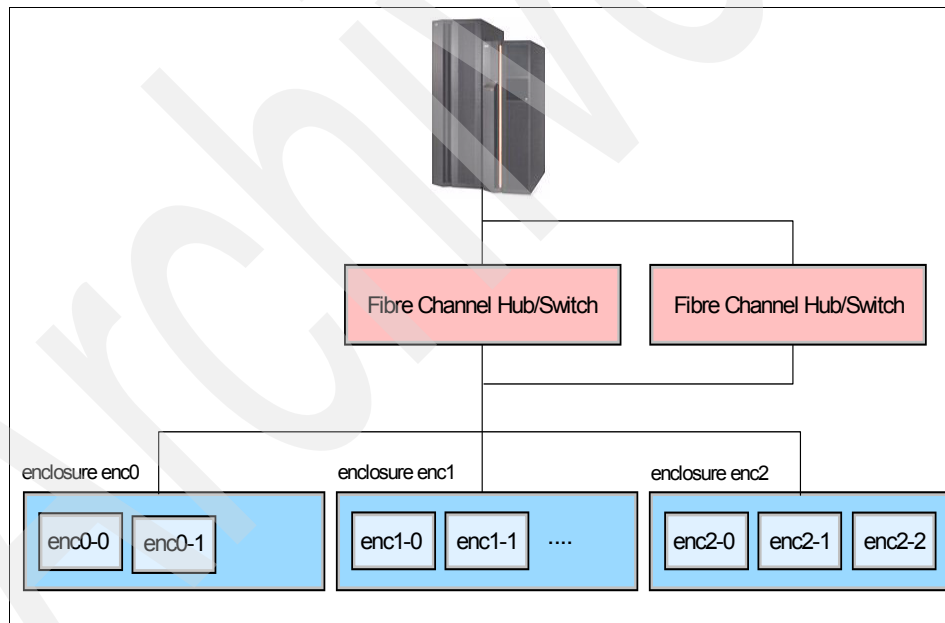


Figure 2-3 Disk enclosures and enclosure-based naming

## 2.2.5 Virtual objects

As mentioned earlier, virtual objects are linked to the physical disks. The virtual objects are:

- ▶ VM disks
- ▶ Disk groups
- ▶ Subdisks
- ▶ Plexes
- ▶ Volumes

We discuss each of these objects in detail in this section. A quick overview of how VxVM uses these objects is as follows:

- ▶ We first link VM disks to the physical disks.
- ▶ The VM disks are aggregated into disk groups.
- ▶ Volumes are then created from the disk groups.
- ▶ Subdisks and plexes are used to further subdivide volumes in order to gain more precise control over data, and to do certain storage management tasks such as striping, mirroring, and RAID 5.

We now discuss each of these objects in more detail.

### VM disks

When a physical disk is to be controlled with VxVM, a VM disk is assigned to that physical disk. A VM disk is a VxVM object that represents a physical disk drive or LUN that is under the control of VxVM. There is a one-to-one correspondence between physical disks and VM disks. Note, that when storage is allocated from the physical disk device, it is allocated from a contiguous area. See more information on this in “Subdisks” on page 21.

Each VM disk has a unique name, which is usually of the form `disk##`. This is the default naming style for VM disks within the root disk group `rootdg`. For other disk groups, the name of each VM disk is usually of the form `diskgroup##`. Note that the default VM disk naming conventions are different depending on the utility used to add the disks. For example, VM disks added to the root disk group `rootdg` by `vxinstall` use the convention `disk##`, but disks added by `vxchg` or `SMIT` use disk access names as default disk media names. Finally, disks added by `VEA` use `diskgroup##` by default for VM disk names. The user can also define whatever name he or she wants for each VM disk. See Figure 2-4 on page 20 for an example of a VM disk named `disk01` in the root disk group `rootdg`, which is controlling physical disk `hdisk1`.

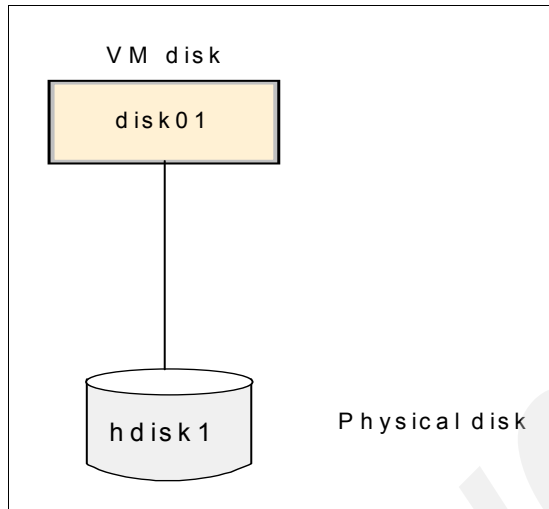


Figure 2-4 VM disk

## Disk groups

A disk group is simply a set of VM disks. By creating disk groups, you can then group sets of VM disks into logical units. The storage capacity that comprises a single volume must be allocated from within a single disk group. See Figure 2-5 for an example of a disk group.

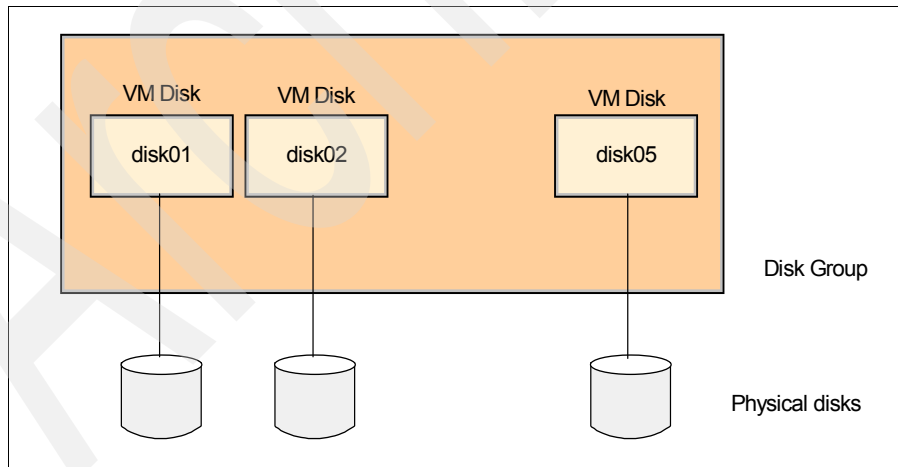


Figure 2-5 Disk group containing three VM disks

## Subdisks

A VM disk can be divided into subdisks. Each subdisk of a VM disk represents a contiguous block of physical disk space. As described above, each VM disk has a name, in the case of the root disk group, usually of the form disk## (for example, disk01). The default name for a subdisk is then disk##-##. For disk groups other than the root disk group, the default name for a subdisk follows the convention disk\_media\_name-##, which by default is usually in the form diskgroup##-##. Again, the user can define any subdisk name that he or she wants. In Figure 2-6, disk01-01 is the name of the first subdisk on the VM disk named disk01. The other two subdisks are disk01-02 and disk01-03.

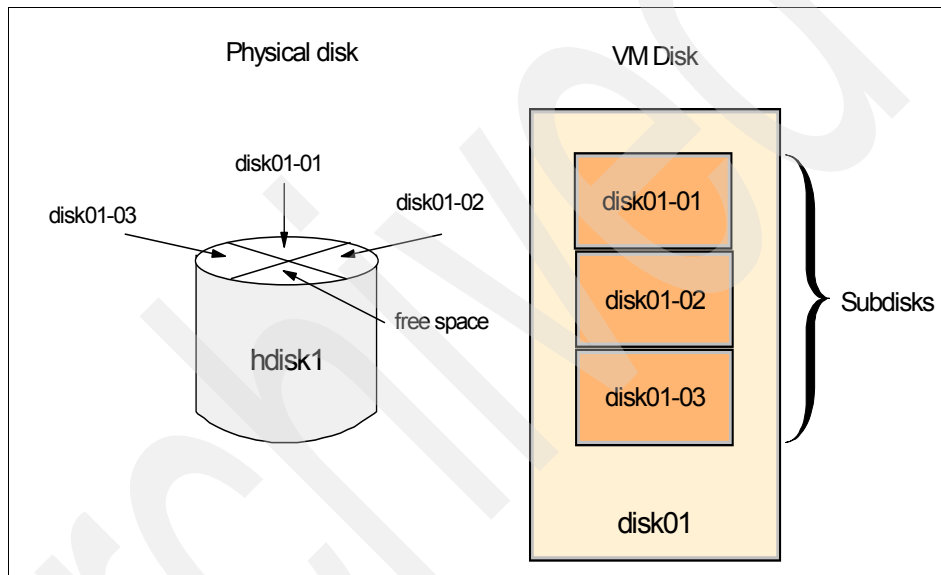


Figure 2-6 Subdisks

Note that the subdisks in a given VM disk do not overlap. Each subdisk is assigned a unique and contiguous portion of the physical disk space controlled by the VM disk. Also, if there is any disk space that has not been assigned to a subdisk, this is simply free space.

## Plexes

We now introduce the concept of a plex. A plex is a set of subdisks. The subdisks in a plex can be located on one or more physical disks. The subdisks do not have to all be from the same physical disk, although they do need to belong to the same disk group. By forming plexes, you now have the power to organize and maintain your storage in many ways, using methods such as striping (RAID 0), mirroring (RAID 1), mirroring with striping (RAID 1+0), striping with mirroring (RAID 0+1), and striping with parity (RAID 5). See Figure 2-7 on page 22 for an

example of a plex that contains two subdisks (disk01-01 and disk 01-02). In this example, as we can see from the default naming convention, both the subdisks come from the same physical disk, but this is not necessary.

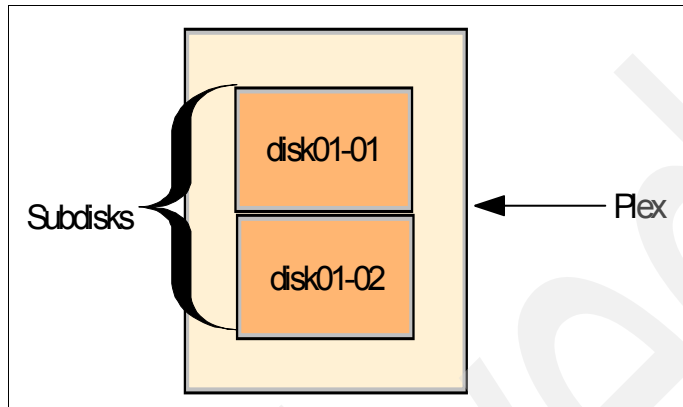


Figure 2-7 Plex with two subdisks

## Volumes

We have covered the following objects so far:

- ▶ The physical disk.
- ▶ The VM disk: A logical element associated with a given physical disk.
- ▶ The disk group: A set of VM disks.
- ▶ The subdisk: A subset of a VM disk, associated with contiguous physical disk space.
- ▶ The plex: A set of subdisks that can be from one or more physical disks.

We now put all these objects together to define the volume. The volume is the main (and only) object that is exported for application use. Each volume consists of one or more plexes. In turn, each plex contains one or more subdisks that are from a single disk group. The disk group is a set of VM disks that are associated with the actual physical disks.

As stated earlier, the volume is a virtual object. The volume appears to applications like a physical disk device. It is not constrained, though, by the physical limitations of a physical disk device. A volume is comprised of one or more plexes, which each contain subdisks that can be from different physical disks, as long as they are from the same disk group. Also, note that a volume must have at least one plex, and, in fact, commonly, a volume is defined with a single plex. Once you have defined a volume, you can associate a file system with that volume. See Figure 2-8 on page 23 for an example of a file system

formatted on a volume (vol01). The volume, vol01, contains a single plex (vol01-01). The plex contains four subdisks: disk01-01, disk01-02, disk01-03, and disk02-01. The subdisks are named according to the VxVM default naming convention, so subdisk disk02-01 is from a different disk than the other three subdisks.

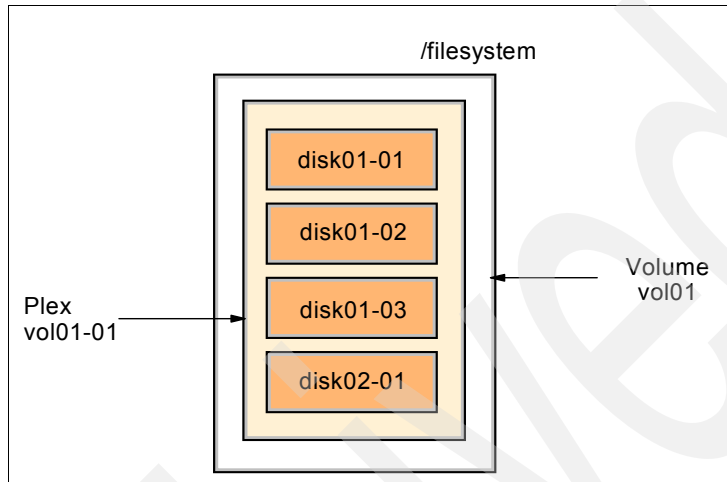


Figure 2-8 File system associated with volume

In Figure 2-9 on page 24, we show how the volume, its plex, and the subdisks relate to the physical disks, disk groups, and VM disks. In the figure, we see the same volume as in Figure 2-8, but also with its corresponding VM disks, disk groups, and physical disk devices. The example again uses default VxVM names. There are two physical disks, `hdisk1` and `hdisk2`. `hdisk1` has three subdisks defined: `disk01-01`, `disk01-02`, and `disk01-03`. `hdisk2` has two subdisks defined: `disk02-01` and `disk02-02`. There are two VM disks, `disk01` and `disk02`, which are in a disk group and control the two physical disks. There is one volume defined called `vol01`. It contains a single plex, `vol01-01`. The plex, `vol01-01` contains four subdisks: `disk01-01`, `disk01-02`, `disk01-03`, and `disk02-01`. Three of the subdisks come from one VM disk and one subdisk comes from another VM disk.

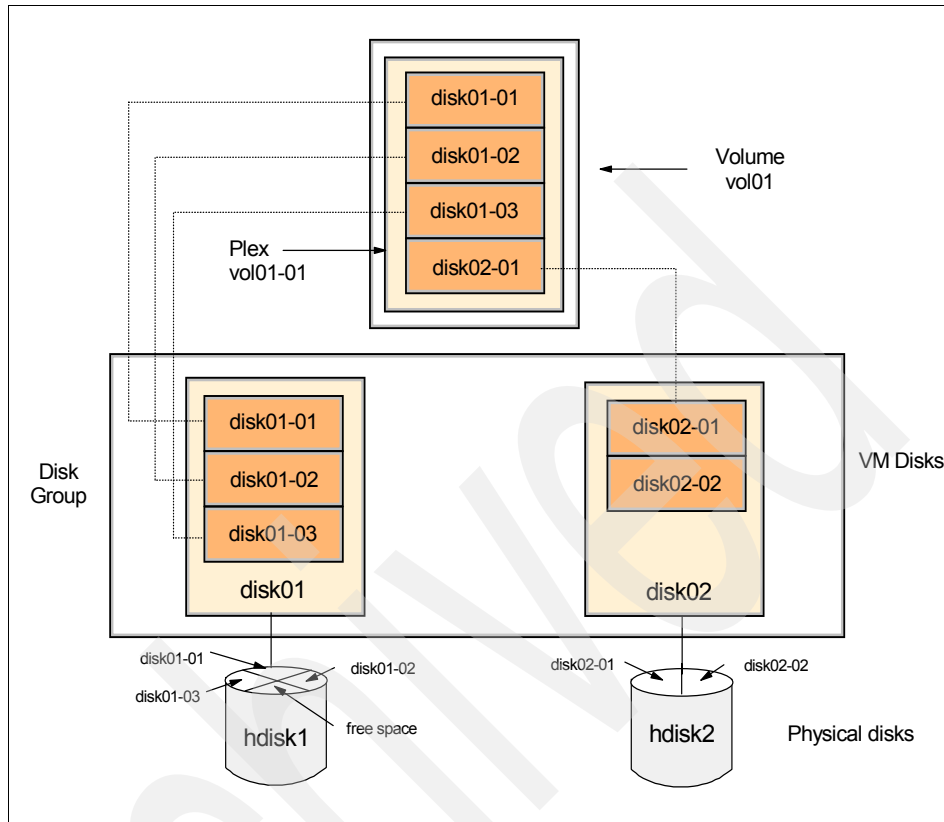


Figure 2-9 Volume with one plex

## 2.2.6 Volume layouts

Different types of application needs require different types of volume layouts. To lay out volumes (and the underlying plexes and subdisks), you may want to use one of the following RAID (Redundant Array of Independent Disks) volume layouts. Using RAID provides a way to improve system I/O performance and data reliability.

We describe these RAID volume layouts in the following sections:

- ▶ Striping (RAID 0)
- ▶ Mirroring (RAID 1)
- ▶ Striping with Mirroring (RAID 0+1)
- ▶ Mirroring with Striping (RAID 1+0)
- ▶ Striping with Parity (RAID 5)



## Concatenation

Before we begin with the introduction to the RAID volume layouts, we first discuss the simple concept of concatenation.

Concatenation is a method where data is laid out in a linear manner from one subdisk to the next in a given plex. Data is put on the first subdisk until it is filled, and then data continues to the next subdisk. The subdisks themselves do not need to be physically contiguous and can belong to different VM disks.

This is a simple volume layout method, as compared with the RAID layouts described in the next sections.

## Striping (RAID 0)

Striping, also known as RAID 0, writes data to the disk device such that the data is distributed evenly over two or more disks. Striping provides higher throughput through I/O load balancing.

How is striping accomplished in VxVM? In VxVM, each volume is made up of at least one plex. The plex is what gets striped. A striped plex contains two or more subdisks which are on two or more physical disks. When the plex of a particular volume is striped, the data is allocated in stripes to the different subdisks which reside on different physical disks. Data is striped across the subdisks that comprise the single plex that makes up a striped volume. The size of the stripes, or the stripe unit size, is determined by the user. In VxVM, the default stripe unit size is 64 kilobytes. Each stripe is a set of contiguous blocks on a disk. See Figure 2-10 on page 26 for an example of striping. In the figure, we have striped the plex, vol01-01, which resides in volume, vol01. When we stripe the plex, we have striped the subdisks within: disk01-01, disk01-02, disk01-03, and disk02-01. The data is then allocated in stripes on the physical disks.

For simplicity, we do not show that the subdisks in VxVM can be grouped into columns. Each column can contain one or more subdisks. See the *VERITAS Volume Manager 3.2 - Administrator's Guide (AIX)* for more information on using columns. If you are unsure of how many columns to use, you can use two columns for a simple striped volume layout.

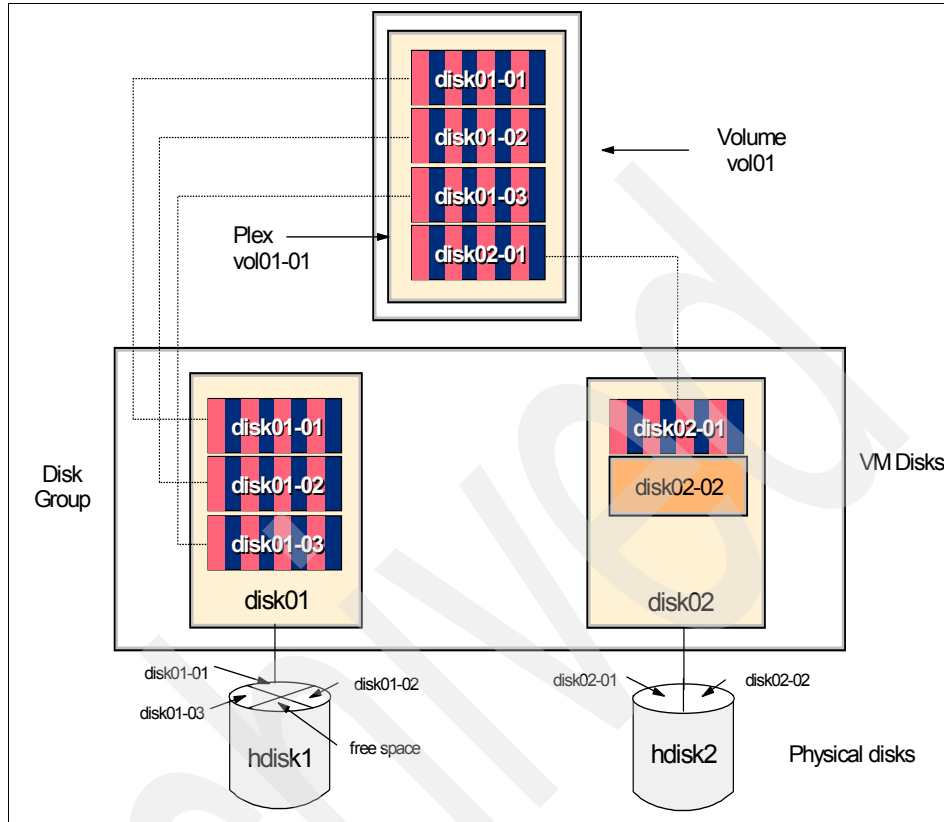


Figure 2-10 Striping of a volume and its associated plex

The advantage of striping that is higher I/O performance is gained because of load balancing over several disks. But striping has a significant disadvantage because it lacks any redundancy. If any single disk device fails, then the entire RAID volume becomes unusable, since the data is spread out over multiple disks. In fact, striping carries a higher risk than using a regular disk with no striping. This is because with more disks a failure on any one disk, is a failure for the entire set of disks. So striping can provide higher I/O throughput but it lacks data reliability.

### Mirroring (RAID 1)

Mirroring, or RAID 1, is a layout which copies (or mirrors) data from one disk drive to another. The mirrored data is a copy of the data from one disk drive onto another disk drive.

Commonly, you use one plex per volume, but to implement mirroring, you need to have at least two plexes: the original plex containing your data and the

mirrored plex, which has a copy of the data. The mirrored plex must mirror to a different physical disk in order to provide data redundancy. If there is a physical disk failure of the original plex, then the mirrored plex still provides access to that same data on another physical disk. In Figure 2-11, we have an example of mirroring where we have a plex, vol01-01, which we mirrored to a plex, vol01-02. Both the plexes are in the same volume, vol01. The mirrored plex does not have to contain the same number of subdisks as the original plex that it is mirroring. In Figure 2-11, plex vol01-01 has two subdisks, and the mirrored plex, vol01-02, has three subdisks. The mirrored plex's subdisks must simply be large enough to accommodate the data contained in the subdisks in the original plex, vol01-01.

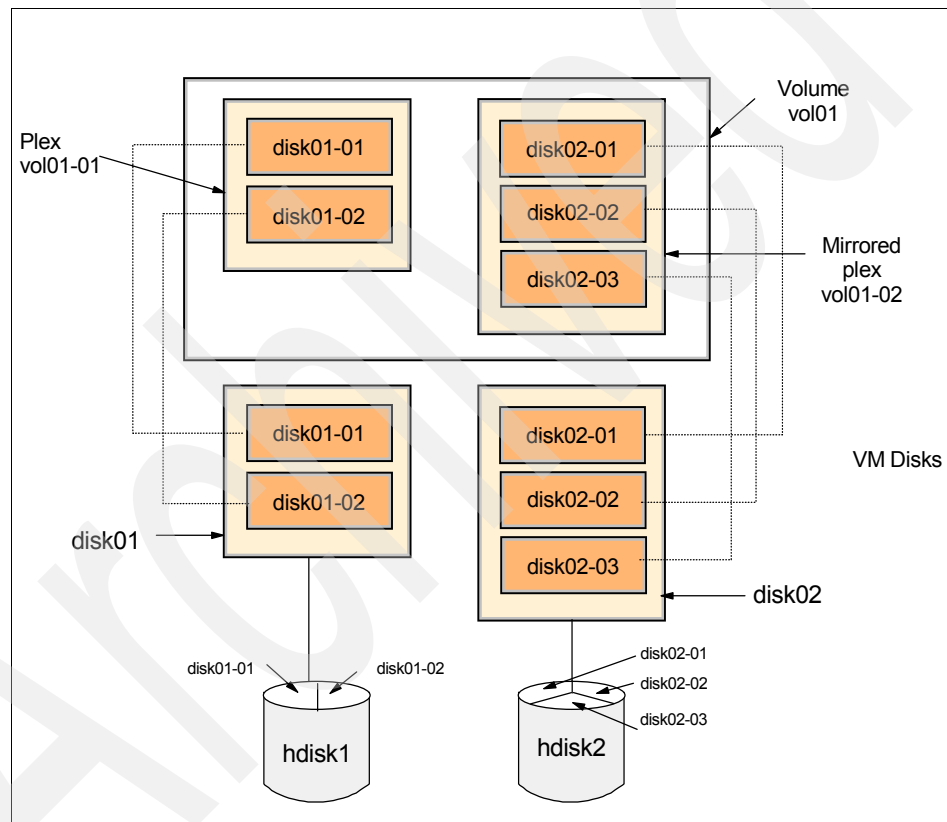


Figure 2-11 Mirroring a plex to plex

Unlike striping, where a single disk failure makes any data unavailable, mirroring does provide data redundancy. This is the main advantage of mirroring. As long as at least one mirror remains usable, your data is available. Mirroring simply requires at least one other disk drive to mirror the data.

## Striping with mirroring (RAID 0+1)

Looking at the advantages of striping, which gives us load balancing, and the advantages of mirroring, which provides data reliability, it would be beneficial to combine the two advantages and do striping with mirroring. If we put the stripes above the mirrors, then this is called RAID 0+1. This method is supported by VxVM.

Putting striping above mirroring essentially creates a mirror of each stripe. See Figure 2-12 for an example of a RAID 0+1 layout. We stripe the disks first in this method, and then mirror each stripe. In the figure, we first striped the plex, vol01-01, and its subdisks, disk01-01 and disk01-02. We then mirrored the plex and its subdisks to plex vol01-02 and its subdisks, disk02-01, disk02-02, and disk02-03. We can see the stripes go through to the underlying VM disks and therefore onto the physical disks themselves.

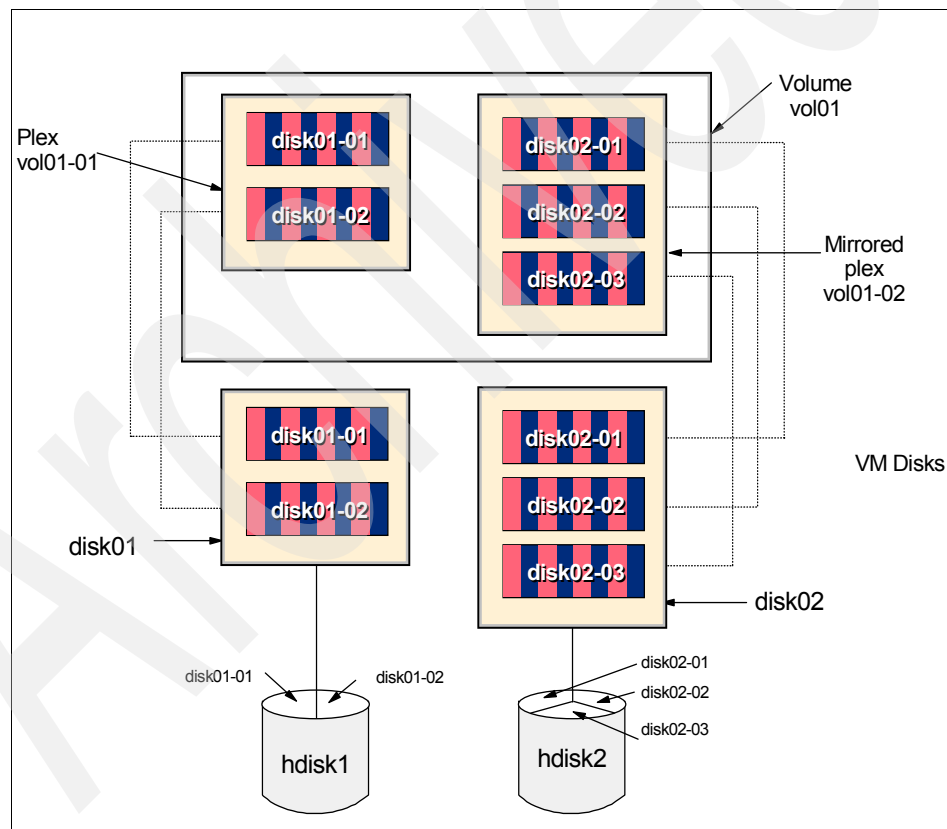


Figure 2-12 RAID 0+1 layout: striping with mirroring

If a physical disk fails in this volume layout, only that portion of the volume loses data redundancy. When the disk is replaced, only a portion of the volume needs to be recovered. Striping with mirroring offers fault tolerance and data reliability from the mirroring, plus the performance benefits of load balancing from the striping.

### **Mirroring with striping (RAID 1+0)**

Another way to combine the advantages of mirroring and striping is to do mirroring with striping. This is called RAID 1+0, or RAID 10. This method is also supported by VxVM.

In RAID 1+0, we mirror the data first, and then we can also stripe the mirror to gain the benefits of spreading the data across several subdisks. In RAID 1+0, we are striping the mirrors, and in RAID 0+1 we are mirroring the stripes. See Figure 2-13 on page 30 for an example where the mirror is striped. Here we have a plex, vol01-01, which we mirror to plex, vol01-02. We then stripe the mirrored plex in this example. We could have also striped the original plex, vol01-01. RAID 1+0 uses a concept called *layered volumes*. See 2.2.7, “Layered volumes” on page 33 for a description of layered volumes.

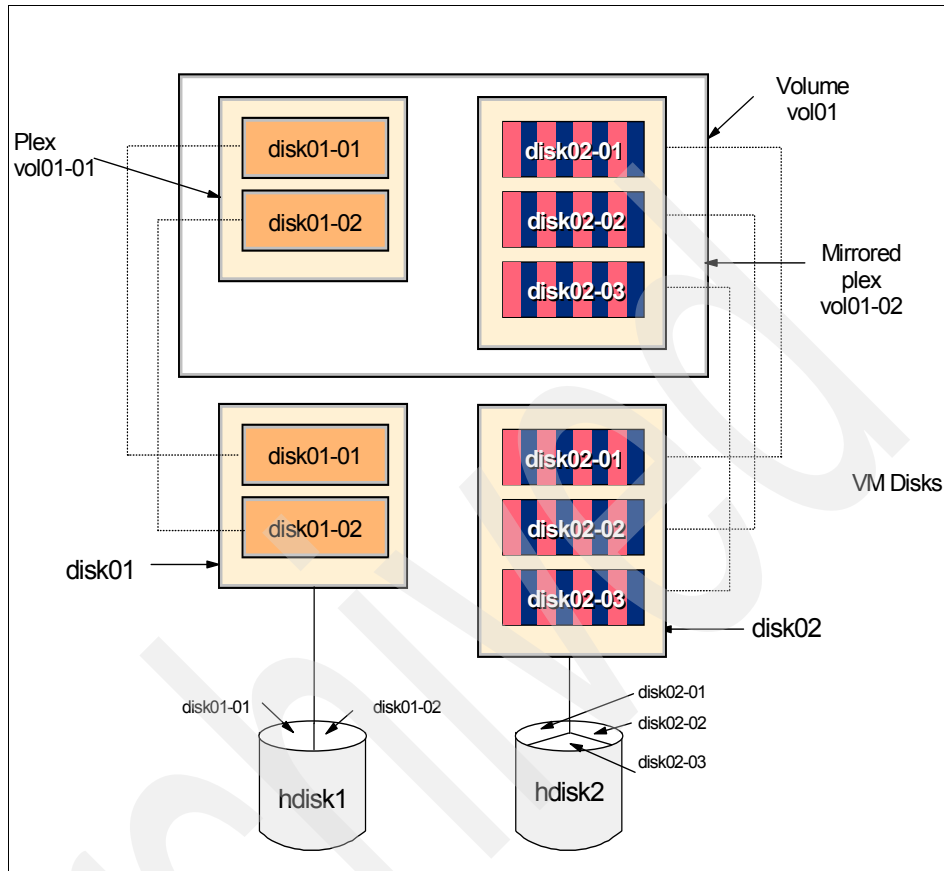


Figure 2-13 RAID 1+0 layout: mirroring with striping

Although here we have contrasted RAID 1+0 and RAID 0+1, the terms RAID 1+0 and RAID 0+1 are often confused in the literature and used interchangeably since they both do mirroring and striping. In the way that we have interpreted RAID 1+0, the data and the mirror are not always both striped, unlike RAID 0+1. We next look at another RAID layout: RAID 5.

### Striping with parity (RAID 5)

Like striping (RAID 0), RAID 5 uses striping but also provides data redundancy. RAID 5 volumes stripe the subdisks in a plex, just as with RAID 0, but one of the stripe units in each stripe contains parity information. This parity information is used to rebuild data if there is a disk failure. Parity is calculated by doing an exclusive OR (XOR) when data is written to a volume. When data is modified, the parity is recalculated and rewritten to the disk. The stripe that contains the parity information is written to a subdisk on a different physical disk than where the data

resides. Because of this requirement, in order to implement RAID 5, a minimum of three physical disks are required. Since the parity data is not stored on the same disk drive as the data it protects, parallel read and write operations are possible, which can increase performance. Write operations will typically access one data drive and one parity drive. However, because the parity is written to a different disk drive, write operations can usually be overlapped.

If there is a disk failure, the data from the failed disk can be reconstructed from the parity information. The data recovery time is relatively slow, but reliable. Every time there is an access to data on the failed disk, the original data needs to be computed using the parity information. Also, reconstructing the storage capacity of the subdisk(s) on the failed disk onto a replacement disk is also a fairly time-consuming process.

In Figure 2-14 on page 32, we have an example of a RAID 5 volume layout. In this example, we have a volume, vol01, containing a single plex, vol01-01. The plex, vol01-01, has three subdisks that are on three different VM disks (and physical disks). The plex is striped, but each striped subdisk contains one stripe unit that has parity information. Subdisk disk01-01 has a parity stripe unit called parity1. Similarly, subdisks disk02-01 and disk03-01 have parity stripe units parity2 and parity3 respectively. On the VM disks (and therefore the physical disks), the parity stripe unit for subdisk disk01-01 resides on VM disk disk02. The parity stripe unit for subdisk disk02-01 resides on VM disk disk03. And finally, the parity stripe unit for subdisk disk03-01 resides on VM disk disk01. The parity stripe unit for a given subdisk resides on a different physical disk. In this example, if there is a disk failure of the physical disk associated with VM disk disk01, then the data can be recovered through the parity information that is still available on VM disk disk02.

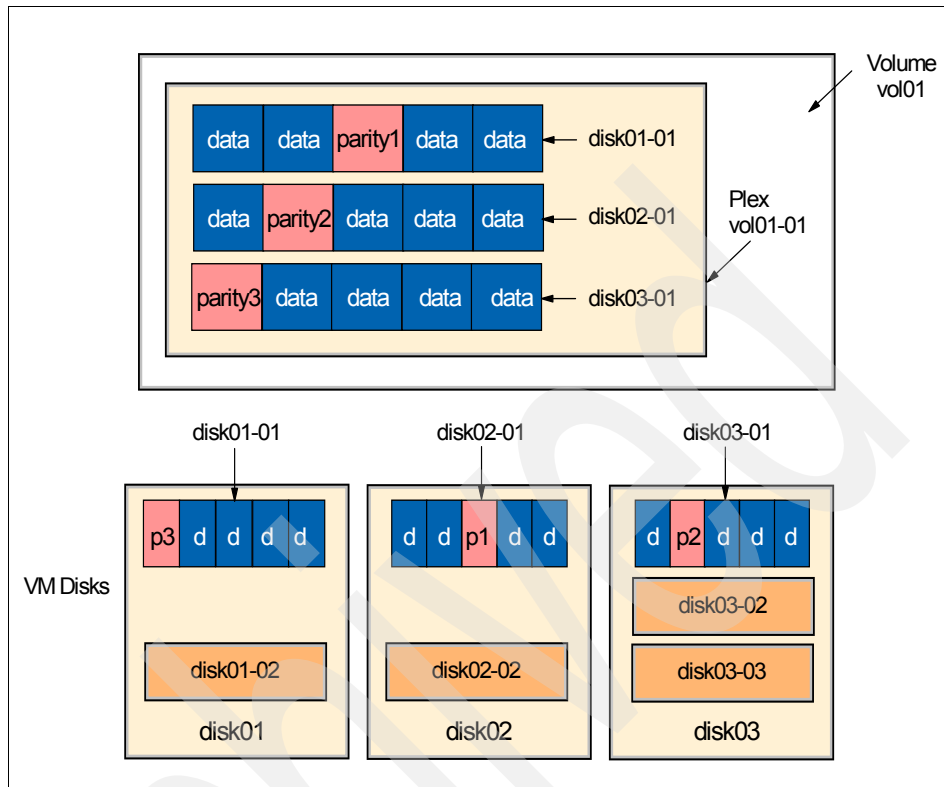


Figure 2-14 RAID 5 layout

### RAID 5 logging

Logging with RAID 5 can speed up data recovery time in case of a disk failure and also improve data reliability. The use of logging is optional. A log is written to a persistent device with the new data and the parity information. This log is written before the data and parity is written to the disk array. This way, if there is a disk failure during a write, but before the data and parity information are actually written to the disks, the information is available in the log of the intent to write the data, and the data can be recovered. You therefore gain additional data reliability by using logging. Logs are implemented in VxVM by using an additional log plex associated with a given RAID 5 volume. This log plex is not a RAID 5 volume layout, but it is possible to mirror the log plex for additional data reliability.

RAID 5 is excellent for applications requiring high availability but having fewer writes than reads. If many reads are needed, then a disk failure will make data reads slower as it reconstructs lost data using parity information, and performance will therefore be reduced.



In sum, RAID 5 gives both good performance and data reliability in an efficient manner, by using parity information to protect the data.

## 2.2.7 Layered volumes

A powerful concept within VxVM is the layered volume. Simply put, a layered volume is a volume that is built on top of another (internal) volume. Instead of building the volume and plex with subdisks that are linked directly to the physical disk, we build the volume and plex with subdisks that are linked to another internal volume, which in turn links to the physical disk. That internal volume cannot be manipulated by the user, and is managed fully by VxVM.

## 2.2.8 Online relayout

If you need to change the type of volume layout, you can. The layout can be changed online, and uses minimal temporary space in order to accomplish the relayout. Online relayout can be used to modify stripe information, change RAID 5 to mirroring, change mirroring to RAID 5, and so on. See Chapter 5, “Advanced administration” on page 131 for information on how to accomplish this task.

## 2.2.9 Hot relocation

When a disk or I/O failure occurs, hot relocation automatically detects the failure and starts to relocate the affected subdisks. You do not need to enable hot relocation; it will start on its own and is enabled by default. Hot relocation, though, is only available for RAID 1 or RAID 5 volumes that have data redundancy. If you have a subdisk with a non redundant volume layout, then it will not be hot relocated.

The subdisks are relocated to disks that you have previously designated as spare disks or to free space within the disk group. When you designate spare disks, they must be placed in the disk group as spares and will only be used for hot relocating. If you do not designate any spares, then free space is used. But if there is not enough free space, then the failed subdisk is not hot relocated, and the system administrator is notified via e-mail.

## 2.2.10 Dirty Region Logging (DRL)

Dirty Region Logging (DRL) improves the data recovery time of mirrored volumes when there is a disk or I/O failure. It is optional and not enabled by default. When a failure occurs, DRL makes all the mirrors consistent. DRL increases data availability by reducing the recovery time after a system crash. The logging process itself will add some extra overhead time, but the process to recover after a failure is faster.

How does DRL recover from a failure? It does this by keeping a dirty bit whenever there is a write to a disk region. A region is a logical area of the volume that DRL defines and uses because it is more efficient than using blocks. The dirty bits associated with the writes to the disk regions are kept in a dirty region log. It is also possible to mirror the dirty region log for greater data integrity. The dirty bits remain in the dirty region log until a maximum number of dirty regions is reached. At that time, the first region that was marked dirty is overwritten.

When there is a failure, VxVM recovers the regions that are marked dirty in the dirty region log. DRL saves time by avoiding the syncing of all the mirrors, because now the system knows which regions are dirty and only needs to sync up the dirty regions. Note that DRL does not guarantee the validity of the data, but simply keeps all the mirrors consistent.

## 2.3 VERITAS File System

VERITAS File System (VxFS) is a powerful, high performance journaled file system that provides for fast recovery and excellent data integrity after system failure as well as easy online file system management.

VxFS supports online system backup, allows for growing and shrinking of file systems, has online defragmentation, and other online file system management capabilities. VxFS also uses a uniform and standard disk layout across platforms, which enables you to use VxFS in a heterogeneous environment. VxFS can be managed by using the VEA GUI.

### 2.3.1 Features

VERITAS File System offers the features shown in this section.

#### High availability features

- ▶ Online file system resizing and database tablespace growth: See 2.3.7, “Online file system resizing” on page 39
- ▶ Online snapshot with support of common backup packages
- ▶ Journaling file system using intent logging: See 2.3.6, “Journaling” on page 38
- ▶ Panic-free error limiting for failed file systems

#### High performance and scalability features

- ▶ Extent-based allocation: See 2.3.3, “Extent-based allocation” on page 36
- ▶ Automated performance tuning interface

- ▶ Online file defragmentation and directory optimization: See 2.3.8, “Online defragmentation” on page 39
- ▶ Quick I/O Database Accelerator for VERITAS Database Edition

### **Centralized management**

- ▶ Quota capabilities that control strategic resources
- ▶ Intuitive VEA GUI: See 2.1.3, “VERITAS Enterprise Administrator overview” on page 11
- ▶ Device driver, file system, and database independence

### **Integration**

- ▶ Support of any UNIX File System (UFS) application
- ▶ Full integration with other VERITAS storage applications

## **2.3.2 Disk layout**

The disk layout describes how data is stored on a file system. VERITAS File System has used different types of disk layouts in the past (Versions 1, 2, and 3) and is currently using a Version 4 disk layout. The Version 4 disk layout has support for large files and file systems up to 1 TB in size. In comparison, in AIX 5L's native JFS2, the maximum size of a file is limited only by the size of the file system itself. JFS has a maximum file size of 64 GB.

The VxFS Version 4 disk layout first divides up the file system space into fixed size allocation units when the file system is created. Also, at file system creation, there are two filesets created: the structural fileset, which defines the file system structure, and the primary fileset, which contains user data.

The structural fileset contains the following structural files:

- ▶ Object Location Table File: Referenced from the super-block and shows the location of other structural files.
- ▶ Label File: Encapsulates the super-block and super-block replicas.
- ▶ Device File: Contains device information.
- ▶ Fileset Header File: Contains fileset information, such as the inode list file, the maximum number of inodes allowed, and large file support information.
- ▶ Inode List File: Inode lists for the structural fileset and the primary fileset.
- ▶ Inode Allocation Unit File: Contains the free inode map, extended operations map, and a summary of inode resources.
- ▶ Log File: Contains the address of the file system intent log.

- ▶ Extent Allocation Unit State File: Indicates the allocation state of each allocation unit.
- ▶ Extent Allocation Unit Summary File: Summary information of the extent allocation units.
- ▶ Free Extent Map File: Contains a map of the free extents.
- ▶ Quotas Files: Quotas that track the resources allocated to each user.

In the following sections, we describe the most important of these elements and their corresponding concepts, such as the inode list file, the fileset header file, the intent log, and the log file. We begin with describing extents and block sizes.

### 2.3.3 Extent-based allocation

A file system can allocate space for files within the file system in different ways. Two common methods are:

- ▶ Extents
- ▶ Blocks

An extent is defined as a sequence of contiguous blocks of data within the file system. A file in the VERITAS File System is allocated as a set of extents. The allocation of extents in VxFS can be based on I/O patterns or allocated explicitly by the user. Extent-based allocation provides for fewer larger I/O operations, which are more efficient than many smaller block operations, if one is doing large sequential file accesses. Extent-based allocation is also used by IBM's native JFS and JFS2. VxFS, JFS, and JFS2 use an extent-based allocation method as opposed to the traditional UNIX File System (UFS), which uses a block-allocation method. For example, Solaris' UFS uses a block-allocation method.

Two values define an extent: the length and the address. The length of the extent is measured in terms of file system blocks. The address is the address of the starting block of the extent.

VxFS supports the following file system block sizes:

- ▶ 1 KB: Default block size for file systems up to 8 GB
- ▶ 2 KB: Default block size for file systems between 8 GB and 16 GB
- ▶ 4 KB: Default block size for file systems between 16 GB and 32 GB
- ▶ 8 KB: Default block size for file systems larger than 32 GB

The file system block size is chosen when creating a file system, and cannot be changed later. We recommend that the default values be used when creating your file system. It is possible that large file systems with just a few files may benefit from a larger block size and gain increased file system performance. Using larger block sizes does waste less disk space in file system overhead, but

it consumes more space for files that are not a multiple of the block size. It is best to use the default file system block sizes as specified above, in all other cases.

In general, the extent-based allocation method used by VxFS has improved file system performance over block-based allocation methods by allowing for fewer, larger I/O operations.

### 2.3.4 Inodes

Inodes are data structures that contain information about files in a given file system. Since directories are a type of a file on a UNIX system, inodes exist for directories too. The default inode size in VxFS is 256 bytes, but the inode size can be set when creating the file system.

Each inode stores this type of information on each file:

- ▶ File length in bytes
- ▶ Owner and group IDs
- ▶ Mode and type of file
- ▶ Access privileges
- ▶ Number of links to the file
- ▶ Time of last access and last modification
- ▶ Time of last inode change
- ▶ Pointers to the file's data
- ▶ File allocation information
- ▶ Direct and indirect extent information

Each VxFS inode can reference up to 10 extents. These are called direct extents, as opposed to indirect extents. If all of the direct extents have been used, then two indirect extents are available, which point to addresses of other extents: one with single indirection and the other with double indirection. The indirect extents are an optimal size of 8 KB by default. When using the default 8 KB value, the indirect extent contains 2048 entries.

#### Inode list

As the name implies, an inode list is a list of inodes. An inode list exists for a given fileset. There is one inode in the list for every file in each fileset. To maintain data integrity, the inode list file is referenced by two inodes. The initial inode list contains the inodes first allocated when creating the file system. This is for each fileset in a file system. During file system use, inodes are dynamically allocated and added into the inode list for the filesets.

## 2.3.5 Caching

VxFS allows an application to set cache parameters called advisories. These cache advisories are in memory only and therefore disappear after a reboot. They can be set using **ioct1**. This section discusses two caching advisories: direct I/O and discovered direct I/O. See the *VERITAS File System 3.4 - Administrator's Guide (AIX)* for discussions of other caching advisories.

### Direct I/O

Direct I/O is an unbuffered form of I/O. With direct I/O, the application can bypass the kernel buffer, and move data directly between the disk drive and a user buffer. This feature can give improved performance. Direct I/O CPU usage is equivalent to doing a raw disk transfer because of decreased CPU overhead.

### Discovered direct I/O

Discovered direct I/O is similar to direct I/O. When a file system has an I/O request that is larger than the `discovered_direct_iosz` parameter, direct I/O is used for that request. Discovered direct I/O is best for large I/O requests, which then get processed in an unbuffered manner. I/O requests smaller than the `discovered_direct_iosz` parameter are treated as buffered I/O.

## 2.3.6 Journaling

In 2.3.3, “Extent-based allocation” on page 36, we discussed extent-based allocation, which is a file system allocation method that improves file system performance. We now look at improving data availability after a system failure by using a technique called journaling.

VERITAS File System is a journaled file system, which uses an intent log, or journal. The file system keeps file system structural information, which is called metadata, for each file. In non-journaled file systems, if there is a system failure, then the metadata write may not complete, and a **fsck** or file system check will be necessary. The **fsck** examines every block for correctness, which is a time-consuming operation. In VxFS, before metadata changes are written, they are written to the intent log. Then the metadata write operation is performed. If the system crashes during that write operation, the intent log is scanned to find what file system changes had not completed, and then those changes are added to the metadata structures. This is a much quicker method, since only the intent log needs to be examined instead of all the blocks.

The type of metadata that is logged is the following:

- ▶ Creating a file
- ▶ Linking, both hard and symbolic
- ▶ Making a directory

- ▶ Making a node
- ▶ Removing a file (unlink)

VERITAS File System provides superior data integrity through use of the intent log. AIX's JFS and JFS2 also are journaled file systems that use a journal log and log metadata changes. Both maintain information on changes in a journal, and go through the journal in order to recover data when there is a system failure. Again, this is a fast method, since the relevant information is maintained within a single log.

### 2.3.7 Online file system resizing

When you create a file system, you indicate the size of the file system. Over time, and as data needs change, the size of the file system may become either too small, or sometimes even too big. With VxFS, it is possible to resize the file system while online and without any disruption to users. The file system can be increased in size or decreased, depending on your needs. See 4.3.7, “Resizing file systems” on page 125 for more information on how to do this task. AIX 5L's JFS and JFS2 also offer the ability to grow a file system size online. The ability to increase the file system size has been available for a long time on both JFS and JFS2, but it is not possible to shrink a JFS or JFS2 file system.

### 2.3.8 Online defragmentation

At the first creation of a file system, the free space is aligned efficiently. With the repeated creation, modification, and deletion of files, the free space gets spread out, creating gaps and fragments of free disk space. When the free space has been fragmented, then file system performance can suffer, since it is harder to find a contiguous block of free space to allocate. At a minimum, allocation takes more time, and in the worst case, requests for contiguous allocation can fail even when the file system has enough free space. When the free space is consolidated, large contiguous files and extents can be allocated. Large extents means fewer extra disk accesses to fetch inodes, and therefore better large file performance.

To solve this problem, online defragmentation can be used. VxFS can do a file system defragmentation even when file systems are being accessed by users. The system administrator can choose to regularly schedule an online defragmentation to keep the file system performance optimal. JFS and JFS2 on AIX also offer the online defragmentation capability.





# Planning and installation

In this chapter, we discuss some planning issues, such as software levels and hardware required. We also provide a detailed guide for installing VERITAS File System and VERITAS Volume Manager on AIX 5L Version 5.1 using the different install options.

This chapter provides the following topics:

- ▶ Planning considerations
- ▶ Detailed installation guide
- ▶ Post installation tasks

## 3.1 Pre-installation planning

This section covers the hardware and software required to Install VERITAS File System (VxFS) Version 3.4 and VERITAS Volume Manager (VxVM) Version 3.2 for AIX. It will also cover details such as licensing and planning disk usage. Further information can be found in the following publications:

- ▶ *VERITAS File System 3.4 Installation Guide (AIX)*
- ▶ *VERITAS Volume Manager 3.2 Installation Guide (AIX)*
- ▶ *VERITAS Volume Manager 3.2 Hardware Notes (AIX)*
- ▶ *VERITAS File System 3.4 Release Notes (AIX)*
- ▶ *VERITAS Volume Manager Release Notes (AIX)*

### 3.1.1 Hardware requirements

At the time of writing this redbook, VERITAS Foundation Suite for AIX is supported on the following IBM servers running AIX 5L Version 5.1 with Maintenance level one:

- ▶ RS/6000 servers: B80, F50, F80, H70, H80, M80, S80, S85, and SP2
- ▶ IBM @server pSeries servers: p610, p620, p640, p660, p680, and p690
- ▶ Any future pSeries servers running AIX 5L Version 5.1c or later

If Dynamic Multipathing (DMP) is not used, VERITAS Foundation Suite for AIX supports all disk arrays supported by AIX 5L.

All Just a Bunch of Disks (JBODs) are supported by VERITAS Foundation Suite for AIX.

#### **IBM storage devices**

VERITAS Foundation Suite with Dynamic Multipathing (DMP) is supported on the following IBM storage devices:

- ▶ IBM E10 models
- ▶ IBM F10 models
- ▶ IBM E20 models
- ▶ IBM F20 models
- ▶ IBM 7133 SSA/FC

## Non-IBM storage devices

VERITAS Foundation Suite with Dynamic Multipathing (DMP) supports the following non-IBM storage devices attached to IBM servers:

- ▶ EMC Symmetrix 3000 Series and 8000 Series
- ▶ Hitachi 5800 Series, 7700E, 9200, 9910, and 9960
- ▶ Sun Microsystems T3

The latest hardware compatibility list for VERITAS Foundation Suite for AIX can be found at:

[http://www.veritas.com/enabled/compatibility\\_lists.html](http://www.veritas.com/enabled/compatibility_lists.html)

### 3.1.2 Operating system and software requirements

The minimum operating system level required for AIX to install VxFS and VxVM is AIX 5L Version 5.1 with maintenance level one. To determine if your system is at this level, run the following command:

```
# instfix -i | grep AIX_ML
All filesets for 5.0.0.0_AIX_ML were found.
All filesets for 5.1.0.0_AIX_ML were found.
All filesets for 5.1.0.0_AIX_ML were found.
All filesets for 5100-01_AIX_ML were found.
All filesets for 5100-02_AIX_ML were found.
```

The last line indicates that maintenance level two is installed. Alternatively, the following command can be run:

```
# oslevel -r
```

When run on a system with AIX 5L Version 5.1 at maintenance level one, the following output will display:

```
5100-01
```

For AIX 5L Version 5.1 at maintenance level two, the following output is shown:

```
5100-02
```

There are also some required APARS. At AIX 5L Version 5.1 with maintenance level one, the following APARS are required:

- ▶ IY24856
- ▶ IY28484
- ▶ IY29813
- ▶ IY29731

For 5L Version 5.1 with maintenance level two, the following APARS are required:

- ▶ IY29813
- ▶ IY29731

To determine whether these APARS are installed, run the following command:

```
# instfix -ivqk "IY24856 IY28484 IY29813 IY29731"
```

This will give the following output:

IY29813 Abstract: cp and mv VxFS extent attributes changes

Fileset bos.rte.commands:5.1.0.26 is applied on the system.

IY29731 Abstract: cpio doesn't copy extended attributes from VxFS file system

Fileset bos.rte.archive:5.1.0.26 is applied on the system.

The above output shows an example of the command being run on a system with maintenance level two installed, so there will be no output details for the first two APARS.

Once the appropriate Maintenance levels and APARS are installed, refer to Table 3-1 to check that the appropriate filesets are at the correct levels. For example, to check the bos.rte.libc fileset, run the following command:

```
# lspp -l bos.rte.libc
Fileset                      Level  State      Description
-----
Path: /usr/lib/objrepos
  bos.rte.libc                5.1.0.25 COMMITTED  libc Library
```

Table 3-1 Minimum fileset levels required for installation

| Fileset   | Required minimum level | Additional notes   |
|-----------|------------------------|--|
| bos.64bit | 5.1.0.25               | Only required if 64 bit is enabled. If /unix is a symbolic link to /usr/lib/boot/unix_64, then this fileset needs to be installed. |
| bos.mp64  | 5.1.0.25               | Only required if 64 bit is enabled. If /unix is a symbolic link to /usr/lib/boot/unix_64, then this fileset needs to be installed. |

| Fileset            | Required minimum level | Additional notes   |
|--------------------|------------------------|--|
| bos.mp             | 5.1.0.17               | Only required on Multiple processor systems. This can be determined running the <b>lsdev -Cc processor</b> command.    |
| bos.up             | 5.1.0.17               | Only required on Systems with one processor. This can be determined by running the <b>lsdev -Cc processor</b> command. |
| bos.rte.archive    | 5.1.0.16               | This is the minimum level required; however, installing APAR IY29731 will update this fileset to 5.1.0.26.             |
| bos.rte.commands   | 5.1.0.16               | This is the minimum level required; however, installing APAR IY29813 will update this fileset to 5.1.0.26.             |
| bos.rte.filesystem | 5.1.0.17               |  |
| bos.rte.libc       | 5.1.0.17               |  |

### 3.1.3 File system space

Checking available file system space is important to avoid running out of space, which causes installation failures. Even though the install process will attempt to increase a file system, this sometimes fails. Refer to Table 3-2 for the required available space in each file system or directory.

Table 3-2 Required file system space

| File system | Required space (Mb) |
|-------------|---------------------|
| /usr        | 130                 |
| /opt        | 143                 |
| /etc        | 1                   |
| /sbin       | 4                   |
| Total       | 278                 |

### 3.1.4 Licensing

Before installing any VERITAS Foundation Suite for AIX products, ensure you have a valid license key available for the products that are going to be installed. License keys can be obtained online by going to <http://vlicense.veritas.com> or by filling out the license key request form that comes with the VERITAS Foundation Suite for AIX package. Further information on VERITAS product licensing can be obtained by sending an e-mail to [license@veritas.com](mailto:license@veritas.com).

### 3.1.5 Selecting disks for use in VxVM

Before starting installation of VERITAS Foundation Suite, determine which disks are attached to your system and which ones are going to be used as VERITAS Volumes. The **lspv** command can be used to list available disks attached to your system. In order to bring a disk under control of VxVM, it must be removed from control of AIX's Logical Volume Manager (LVM). Use the **chpv** command to remove a disk from LVM control. For example, to remove hdisk4 from LVM control, run the following command:

```
# chpv -C hdisk4
```

The VxVM installation and initialization process will not touch any disk that is part of a LVM volume group and will not be able to initialize any disk that is not removed from control of the LVM via the **chpv** command.

As the current version of VxVM does not support encapsulation of the root file systems, at least one extra disk is required by VxVM. When installing VxVM, it will create a primary disk group called the rootdg (not to be confused with LVM's rootvg), in which at least one disk needs to be placed into.

**Note:** The primary disk group in VERITAS Volume Manager is called the rootdg. This might cause some confusion with LVM's rootvg. Future releases of VERITAS Volume Manager will not have rootdg as the primary disk group.

Another consideration before starting installation is to check what disk enclosures are attached to your system and more importantly how many paths there are to the disks attached to the system, as the **vxinstall** process will give you the option of specifying which paths to use to the disks if they are multipathed. Typically, the path to a disk is via a controller.

## 3.2 Installation

In this section, we introduce the three installation options of VxFS and VxVM and for each option we step through the install process. The installation files can be

found on the VERITAS Foundation Suite for AIX Installation CD or, alternatively, there is an evaluation copy on the AIX 5L Version 5.1 Bonus Pack CD. The three installation options are:

- ▶ Using the VERITAS supplied VRTSinstall script
- ▶ Using SMIT
- ▶ Using **installp**

For all three options the VERITAS Foundation Suite for AIX CD is required to be mounted or the installation files be accessible on the host you are installing to. Place the CD in the CD ROM drive and, assuming /dev/cd0 is the path to the CDROM drive, run the following command:

```
# mount -V cdrfs -o ro /dev/cd0 /cdrom
```

Once the CD is mounted, choose one of the following three installation options:

- ▶ Section 3.2.1, “Installation using VRTSinstall” on page 48
- ▶ Section 3.2.2, “Installation using SMIT” on page 49
- ▶ Section 3.2.3, “Installation using installp” on page 52

Regardless of what installation method is chosen, the installation should be run as the root user ID.

Possibly the best option for installation would be to use SMIT, as this will keep a log file of the installation in the smit.log (located in the root users home directory) and will install the packages in the correct order, provided the install requisite software option is set to yes. Table 3-3 shows a list of VxFS and VxVM packages. When installing these packages, the correct order should be to install VRTSvlic first, followed by VRTSvxfs and VRTSvxvm. If installing the VEA, the order of install should be VRTSob, VRTSobgui, VRTSfspro, and VRTSvmpro.

*Table 3-3 VxFS and VxVM packages*

| Package   | Description                                     |
|-----------|---|
| VRTSvlic  | VERITAS product licensing                       |
| VRTSvxfs  | VERITAS File System                             |
| VRTSfsdoc | VERITAS File System Documentation - Optional    |
| VRTSvxvm  | VERITAS Volume Manager                          |
| VRTSvmman | VERITAS Volume Manger man pages - Optional      |
| VRTSvmdoc | VERITAS Volume Manager documentation - Optional |

| Package   | Description  |
|-----------|--|
| VRTSob    | VEA Service - Optional, only required if using VEA                             |
| VRTSobgui | VEA java console - Optional, only required if using VEA                        |
| VRTSfspro | VERITAS File System provider package - Optional, only required if using VEA    |
| VRTSvmpro | VERITAS Volume Manager provider package - Optional, only required if using VEA |

For a minimum installation of VxVM and VxFS, the VRTSvlic, VRTSvxfs, and VRTSvxvm packages are required.

### 3.2.1 Installation using VRTSinstall

The most popular way is to use the VRTSinstall script as follows:

- Change to the /cdrom/pkgs directory or if installing off the AIX Bonus pack CD, change to the /cdrom/other/VRTSFST/pkgs directory.
- Run the ./VRTSinstall script. The screen shown in Example 3-1 will appear.

*Example 3-1 VRTSinstall main window*

```
*****
Veritas Foundation Suite Installation
*****
```

1. VRTSfsdoc
2. VRTSfspro
3. VRTSob
4. VRTSobgui
5. VRTSvlic
6. VRTSvmdoc
7. VRTSvmman
8. VRTSvmpro
9. VRTSvxfs
10. VRTSvxvm

```
Select package(s) you wish to process (or 'all' to process all packages).
(default: all) [(1-12),?,??,q]:
Select the Packages to install
```



**Note:** At the time of writing this redbook, when running VRTSinstall on a 32 bit operating system that does not have the bos.64bit and bos.mp64 filesets installed, the following syntax error may appear at the top of the screen:

```
/usr/bin/ls1pp: 0504-132 Fileset bos.64bit not installed.  
/usr/bin/ls1pp: 0504-132 Fileset bos.mp64 not installed.  
./VRTSinstall[21]: test: 0403-004 Specify a parameter with this command.  
./VRTSinstall[21]: test: 0403-004 Specify a parameter with this command.
```

This does not hinder the functionality of VRTSinstall.

Depending on the CD, the package options may vary. Select a package to install by entering the number to the left of the package name. When that package has finished installing, select the next package to install.

**Note:** At the time of writing this redbook, there is a syntax error with the VRTSinstall script that does not allow multiple packages to be selected on the same line. For example selecting 1,2,4 may give the following error:

```
./VRTSinstall[373]: 1,2,4: 0403-057 Syntax error
```

The default option for VRTSinstall is to install all packages, but there may be some packages that are not required for VxFS and VxVM, or you might not want to install some of the optional packages. See Table 3-3 on page 47 for the packages and order of install. Once all packages are installed, proceed to 3.3, “Post-installation tasks” on page 53.

### 3.2.2 Installation using SMIT

From the command line, run the following command:

```
# smitty install_latest
```

The SMIT directory input screen is shown in Example 3-2.

*Example 3-2 SMIT install directory input screen*

---

|  |                      |
|--|----------------------|
| Install Software   |                      |
| Type or select a value for the entry field.<br>Press Enter AFTER making all desired changes. |                      |
| * INPUT device / directory for software  | [Entry Fields]<br>[] |

---

Type in the path to the installation CD or file system with the installation files, then press Enter. For the VERITAS Foundation Suite for AIX Installation CD, this will be /cdrom/pkg, and for the AIX 5L Version 5.1 bonus pack CD, this will be /cdrom/other/VRTSFST/pkg. The screen shown in Example 3-3 appears.

Example 3-3 SMIT install screen

Install Software

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

\* INPUT device / directory for software

\* SOFTWARE to install

PREVIEW only? (install operation will NOT occur)

COMMIT software updates?

SAVE replaced files?

AUTOMATICALLY install requisite software?

EXTEND file systems if space needed?

OVERWRITE same or newer versions?

VERIFY install and check file sizes?

Include corresponding LANGUAGE filesets?

DETAILED output?

Process multiple volumes?

ACCEPT new license agreements?

Preview new LICENSE agreements?

[Entry Fields]

/cdrom/pkg

[\_all\_latest]

no

yes

no

yes

yes

no

no

yes

no

yes

no

no

F1=Help

F2=Refresh

F3=Cancel

F4=List

F5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do

The cursor should be in the SOFTWARE to install field; if it is not, use the up or down arrow to move it there and then press F4. A list of packages appears, as shown in Example 3-4.

Example 3-4 SMIT install fileset selection screen

SOFTWARE to install

Move cursor to desired item and press F7. Use arrow keys to scroll.  
ONE OR MORE items can be selected.  
Press Enter AFTER making all selections.

[MORE...6]

#-----

VRTSfsdoc

ALL

+ 3.4.2.0 VERITAS File System Documentation

VRTSfspro

ALL

+ 3.4.0.0 VERITAS File System Services Provider

[MORE...29]

F1=Help

F2=Refresh

F3=Cancel

F7=Select

F8=Image

F10=Exit

Enter=Do

/=Find

n=Find Next

Using the arrow keys to move each package to install and press the F7 key. This will place a > next to the package name, indicating it has been selected for installation. To unselect a package, press the F7 key again. The packages required to install VxFS and VxVM are listed in Example 3-5; in this case, all packages can be selected, as there are no additional products to exclude.

*Example 3-5 SMIT fileset selection screen with filesets selected*

|   |     |
|---|-----|
| > VRTSfsdoc   | ALL |
| + 3.4.2.0 VERITAS File System Documentation                         |     |
| > VRTSfspro   | ALL |
| + 3.4.0.0 VERITAS File System Services Provider                     |     |
| > VRTSob  | ALL |
| + 3.0.255.0 VERITAS Enterprise Administrator Service                |     |
| > VRTSobgui   | ALL |
| + 3.0.255.0 VERITAS Enterprise Administrator                        |     |
| > VRTSvlic  | ALL |
| + 2.10.4.0 VERITAS License Utilities                                |     |
| > VRTSvmdoc   | ALL |
| + 3.2.0.0 VERITAS Volume Manager User Documentation                 |     |
| > VRTSvmman   | ALL |
| + 3.2.0.0 VERITAS Volume Manager manual pages (3.2p)                |     |
| > VRTSvmpro   | ALL |
| + 3.2.0.0 Virtual Disk VERITAS Management Services Provider Library |     |
| > VRTSvxfs  | ALL |
| + 3.4.2.0 VERITAS File System (VxFS)                                |     |
| > VRTSvxvm  | ALL |
| + 3.2.0.0 VERITAS Volume Manager (3.2p)                             |     |

F1=Help

F2=Refresh

F3=Cancel

F7=Select

F8=Image

F10=Exit

Enter=Do

/=Find

n=Find Next

Once all packages are selected, press Enter; the screen will then return to the one shown in Example 3-3 on page 50. Check that the options are the same as that shown in this example, particularly the “AUTOMATICALLY install requisite

software” option; this should be set to yes to ensure the order of install is correct. If they are not the same, use the arrow key to move to the appropriate field and press the Tab key to change them. Press enter again to start the install. Once the install is complete, a log file of the install can be viewed in the smit.log located in the root users home directory.

Once the installation is complete, proceed to 3.3, “Post-installation tasks” on page 53.

### 3.2.3 Installation using installp

The following command can be used to install VxFS and VxVM packages from the command line:

```
# installp -acXd /cdrom/pkgs VRTSvlic VRTSvxfs VRTSfsdoc VRTSvxvm VRTSvman \
VRTSvmdoc VRTSob VRTSobgui VRTSfspro VRTSvmpo
```

If installing from the AIX 5L Version 5.1 bonus pack CD, the following command can be run:

```
# installp -acXd /cdrom/other/VRTSFST/pkgs VRTSvlic VRTSvxfs VRTSfsdoc \
VRTSvxvm VRTSvman VRTSvmdoc VRTSob VRTSobgui VRTSfspro VRTSvmpo
```

The -ac option is used to commit the updates. The X option will attempt to resize the file system, should it require more space, and the d option specifies the path to the installation files. To keep a log file of the installation, the command can be redirected to a log file using a redirect or piped through the **tee** command, as shown in Example 3-6 and Example 3-7. The advantage of piping the command to the **tee** command is that it will show the output on the screen as well as write to the log file.

---

#### *Example 3-6 Redirecting installp output to a log file*

---

```
# installp -acXd /cdrom/pkgs VRTSvlic VRTSvxfs VRTSfsdoc VRTSvxvm VRTSvman
VRTSvmdoc VRTSob VRTSobgui VRTSfspro VRTSvmpo > /var/tmp/VRTSinstall.log 2>&1
```

---

---

#### *Example 3-7 Redirecting installp output to a log file using tee*

---

```
# installp -acXd /cdrom/pkgs VRTSvlic VRTSvxfs VRTSfsdoc VRTSvxvm VRTSvman
VRTSvmdoc VRTSob VRTSobgui VRTSfspro VRTSvmpo | tee /var/tmp/VRTSinstall.log
```

---

Once the installation is complete, proceed to 3.3, “Post-installation tasks” on page 53.

## 3.3 Post-installation tasks

In this section, we describe the post installation tasks:

- ▶ Installing product licenses
- ▶ Initializing VERITAS Volume Manager using vxinstall
- ▶ Checking VERITAS Volume Manager daemons
- ▶ Setting up the man pages and PATH variable
- ▶ Starting the VEA server
- ▶ Installing VRTSexplorer (Optional)

### 3.3.1 Installing product licenses

After installation of the required filesets, the next step is to install a license key for VxVM and a license key for VxFS. Follow the procedure below to install these license keys:

Run the following command:

```
# /sbin/vxlicinst
```

A screen as shown in Example 3-8 appears.

*Example 3-8 vxlicinst screen*

---

```
VERITAS License Manager vxlicinst Utility Version 2.10, Build 4  
Copyright (C) VERITAS Software Corp 2002. All Rights reserved.
```

```
Using VERITAS License Manager API Version 2.10, Build 4
```

```
Please enter your license key :
```

---

Type in the license key for VxFS, then press Enter. A screen similar to that in Example 3-9 appears.

*Example 3-9 vxlicinst license install screen*

---

```
VERITAS License Manager vxlicinst Utility Version 2.10, Build 4  
Copyright (C) VERITAS Software Corp 2002. All Rights reserved.
```

```
Using VERITAS License Manager API Version 2.10, Build 4
```

```
Please enter your license key : 63069613697805629648898
```

```
Key successfully installed in
```

```
/etc/vx/licenses/lic/63069613697805629648898.vxlic
```

---

Repeat the above procedure for the VxVM license key. Once all license keys are installed, they can be reviewed by running the following command:

```
# /sbin/vxlicrep
```

This produces output similar to Example 3-10.

*Example 3-10 vxlicrep output*

---

```
VERITAS License Key Reporter version 2.10.004
Copyright (C) VERITAS Software Corp 2002. All Rights reserved.

Creating a report on all VERITAS products installed on this system

-----*****-----

License Key           = 63069613697805629648898
Product Name          = VERITAS File System
Key is                = Valid
License Type          = PERMANENT
Site License          = YES

Features :=
  VXFS                = Enabled

-----*****-----

License Key           = 59642249071562707016955
Product Name          = VERITAS Volume Manager
Key is                = Valid
License Type          = PERMANENT
Site License          = YES

Features :=
  VxVM                = Enabled
```

---

### 3.3.2 Initializing VERITAS Volume Manager

After the filesets required for VxFS and VxVM have been installed, the next step is to initialize the VERITAS Volume Manager using the **vxinstall** utility. This does the following:

- ▶ Reviews licenses
- ▶ Adds a disk or multiple disks to VxVM
- ▶ Sets up the rootdg disk group
- ▶ Starts the required daemons

To initialize the VERITAS Volume Manager, at least one disk must be available to be added to the rootdg disk group. Run the following command to initialize VxVM:

```
# /usr/sbin/vxinstall
```

The screen shown in Example 3-11 appears.

---

*Example 3-11 vxinstall license window*

---

VxVM uses license keys to control access. You must have an installed, valid license key to make use of this software. See documentation for more details.

```
Licensing information:
  System host ID: 000BC6CD4C00
  Host type: IBM,7025-F50
```

Some licenses are already installed. Do you wish to review them  
[y,n,q,?] (default: y)

---

Press n and then press Enter. Then, in response to the following prompt, press n and then Enter:

Do you wish to enter a license key [y,n,q,?] (default: n)

A screen appears which describes the naming conventions used by volume manager for any disk enclosures that it detects, as shown in Example 3-12.

---

*Example 3-12 vxinstall screen*

---

Volume Manager Installation  
Menu: VolumeManager/Install

VxVM uses the following format to name disks on the system:

```
<enclosure name>_<diskno>
```

In the above format, <enclosure\_name> is the logical name of the enclosure to which the disk belongs. VxVM assigns default enclosure names which can be changed according to the user requirements.

Some examples would be:

|            |  |
|------------|--|
| shark0_2   | - second disk detected in enclosure 'shark0'   |
| enggdept_2 | - second disk detected in enclosure 'enggdept' |
| dgc1_1     | - first disk detected in enclosure 'dgc1'      |
| Disk_1     | - first disk detected in the jbod category     |

Disks that are not multipathed by VxVM will be named in hdisk# format.

Hit RETURN to continue.

---

Press Enter; a list of disk enclosures is then displayed on screen. There will be no enclosures listed other than OTHER\_DISKS if there are no external disk enclosures attached to the system; otherwise, it will list all enclosures that it detects. This screen gives you the option of renaming the disk enclosures listed. To rename an enclosure, press Enter and then type in the name of the enclosure that is to be renamed; entering list at this point will list the current enclosures. Press Enter and then type in the new name. Once the new name is typed in, press Enter. Example 3-13 shows an example of renaming the OTHER\_DISKS enclosure to INTERNAL\_DISKS.

---

*Example 3-13 Rename enclosure*

---

Rename an enclosure

Menu: VolumeManager/Install/Rename enclosure

Enter an enclosure name: [list,q,?] list

The following is the list of enclosures attached to your system

| ENCLR_NAME  | ENCLR_TYPE  | ENCLR_SNO   | STATUS    |
|-------------|-------------|-------------|-----------|
| =====       |             |             |           |
| OTHER_DISKS | OTHER_DISKS | OTHER_DISKS | CONNECTED |

Hit RETURN to continue.

Enter an enclosure name: [list,q,?] OTHER\_DISKS

Enter the new name for enclosure OTHER\_DISKS: INTERNAL\_DISKS

The enclosure name OTHER\_DISKS has been changed to INTERNAL\_DISKS

Hit RETURN to continue.

---

Once all enclosures are renamed, or if no enclosures need to be renamed, press Enter. A screen describing the quick installation and custom installation options appears as in Example 3-14.

---

*Example 3-14 vxinstall installation options description*

---

Volume Manager Installation

Menu: VolumeManager/Install

You will now be asked if you wish to use Quick Installation or Custom Installation. Custom Installation allows you to select how the Volume Manager will handle the installation of each disk attached to your system.



Quick Installation examines each disk attached to your system and attempts to initialize the disks not owned by LVM and disks not having mounted file systems.

If you want to exclude any devices from being seen by VxVM or not be multipathed by vxmp then use the Prevent multipathing/Suppress devices from VxVM's view option, before you choose Custom Installation or Quick Installation.

If you do not wish to use some disks with the Volume Manager, or if you wish to reinitialize some disks, use the Custom Installation option. Otherwise, we suggest that you use the Quick Installation option.

Hit RETURN to continue.

---

Press Enter to go past this screen. The installation options screen now appears, as shown in Example 3-15.

*Example 3-15 Installation options*

---

Volume Manager Installation Options  
Menu: VolumeManager/Install

- 1 Quick Installation
- 2 Custom Installation
- 3 Prevent multipathing/Suppress devices from VxVM's view
  
- ? Display help about menu
- ?? Display help about the menuing system
- q Exit from menus

Select an operation to perform:

---

Choose an Installation option by entering the number to the left of the option. For the Quick Installation option, refer to “Quick installation” on page 58. For the Custom Installation Option, refer to “Custom installation” on page 60. Both options will find all disks that are available for use by VxVM, place them under VxVM control, and create the primary disk group (rootdg). **vxinstall** will not use disks that fall into these categories:

- ▶ Disks that have file systems mounted on them
- ▶ Disks that are listed in the /etc/vx/disks.exclude file
- ▶ Disks that are on a controller listed in the /etc/vx/cntrls.exclude file
- ▶ Disks that are in an enclosure listed in the /etc/vx/enclr.exclude file
- ▶ Disks that are under control of the LVM

Below is an example of the /etc/vx/disk.exclude file:

```
# cat /etc/vx/disks.exclude
hdisk2
```

If there are any disks that should not be touched as part of the VxVM install, place them in one of the three files mentioned above. The main difference between the Quick Installation option and the Custom Installation option is the Custom Installation option will prompt for confirmation of each disk to be initialized. The recommended option is the Quick Installation if you are sure that you have the correct disks to exclude listed in any of the exclude files; otherwise, use the Custom Installation option. If there are disks attached to the system that are attached to more than one controller or that have multiple paths to them, select option 3 “Prevent multipathing/Suppress devices from VxVM’s view” if you do not want VxVM to multipath these disks or want to exclude any paths to certain disks with multiple paths. Select option 3 first if any Multipathing or path exclusion changes need to be made before selecting option 1 or 2.

### Quick installation

Select 1 and then press Enter to take the Quick Installation option. The screen shown in Example 3-16 appears.

#### *Example 3-16 Quick installation*

---

```
Volume Manager Quick Installation
Menu: VolumeManager/Install/QuickInstall/INTERNAL_DISKS

Disk array serial number : OTHER_DISKS

Generating list of disks in disk array INTERNAL_DISKS....

<excluding root disk hdisk0>
<excluding hdisk2>

The Volume Manager has detected the following disks in disk array
INTERNAL_DISKS:

    hdisk1

Hit RETURN to continue.
```

---

Example 3-16 shows the screen when the Quick Installation is chosen. Note that it has excluded hdisk0, because it belongs to LVM, and hdisk2, because it is listed in the /etc/vx/disks.exclude file. Press Enter, and the next screen displays a warning about Multipathing, as shown in Example 3-17 on page 59.

### *Example 3-17 Quick installation multipathing warning*

---

Volume Manager Custom Installation

Menu: VolumeManager/Install/QuickInstall/OTHER\_DISKS

NOTE: VxVM does not multipath disks in the OTHER\_DISKS category. Hence, if the above disks are connected in a multipathed fashion, you will see one VxVM disk for every path to the disk. This might lead to data corruption if proper care is not taken while initializing these disks.

If these disks return unique serial number on SCSI inquiry, you can use the command 'vxddladm' with the 'addjbod' option to get them multipathed by VxVM. Please refer to VxVM documentation for more information on 'vxddladm'. You can quit vxinstall from the next screen, run 'vxddladm addjbod vid=<vendor-id returned by disk>' and restart vxinstall.

If these disks don't return unique serial number on SCSI inquiry, you are advised not to connect them in a multipathed fashion.

Hit RETURN to continue.

---

Press Enter after the warning; the screen shown in Example 3-18 appears.

### *Example 3-18 Quick installation confirmation message*

---

Volume Manager Quick Installation For Disk Array INTERNAL\_DISKS

Menu: VolumeManager/Install/QuickInstall/INTERNAL\_DISKS

Initialize all disks in this disk array ? (destroys data on these disks)  
[y,n,q,?] (default: n)

---

Enter y and then press Enter. A warning is displayed about destroying data on disks, as shown in Example 3-18. Press y and then Enter. A message appears indicating the volume manager will initialize all disks in the array that were listed in Example 3-16 on page 58. Press Enter, and the screen shown in Example 3-19 appears.

### *Example 3-19 Quick install rename disks*

---

Volume Manager Quick Installation

Menu: VolumeManager/Install/QuickInstall/INTERNAL\_DISKS/Init

Use default disk names for these disks? [y,n,q,?] (default: y)

---

This gives you the option to rename the VxVM disk that VxVM uses to access the disk from the default of diskxx where xx is 01 to 99. To change the disk name,

press n and then Enter. This goes through each disk and allows the new name to be typed in. Type in the new name for the disk when prompted for each disk. Once all the disks are renamed or, if you do not wish to rename any disks, press Enter, and the screen shown in Example 3-20.

---

*Example 3-20 Quick install choice summary*

---

Volume Manager Quick Installation  
Menu: VolumeManager/Install/QuickInstall

The following is a summary of your choices.

hdisk1 New Disk

Is this correct [y,n,q,?] (default: y)

---

In Example 3-20, hdisk1 will be initialized and added to the rootdg disk group. Press Enter, and the output shown in Example 3-21 will appear.

---

*Example 3-21 Quick install initialization screen*

---

The Volume Manager is now reconfiguring (initializing phase)...

Volume Manager: Initializing hdisk1 as a new disk.

The Volume Manager is now reconfiguring (initialization phase)...

Volume Manager: Adding disk01 (hdisk1) as a new disk.

The Volume Daemon has been enabled for transactions.

---

## **Custom installation**

Select 2 and then press Enter to choose the Custom Installation option. A screen similar to Example 3-16 on page 58 will be displayed; however, it will have Custom Installation instead of Quick Installation as the menu heading. Press Enter to continue, and a warning screen about Multipathing will be displayed, as shown in Example 3-17 on page 59. Press Enter to continue past the warning screen. The screen shown in Example 3-22 appears.

---

*Example 3-22 Custom install*

---

Installation options for enclosure INTERNAL\_DISKS  
Menu: VolumeManager/Install/Custom/INTERNAL\_DISKS

- 1 Install all disks as new disks. (discards data on disks!)
- 2 Install one disk at a time.
- 3 Leave these disks alone.

```
?      Display help about menu
??     Display help about the menuing system
q      Exit from menus
```

Select an operation to perform:

---

Option 1 will do much the same process as taking the Quick Install Option. Select “Install one disk at a time” and then press Enter. The screen shown in Example 3-23 appears for the first disk.

*Example 3-23 Custom install one disk at a time*

---

```
Installation options for disk hdisk1
Menu: VolumeManager/Install/Custom/INTERNAL_DISKS/hdisk1
```

```
1      Install as a new disk. (discards data on disk!)
2      Leave this disk alone.

?      Display help about menu
??     Display help about the menuing system
q      Exit from menus
```

Select an operation to perform:

---

Here we have the option of adding the disk to the list to be initialized or leaving it alone. Select option 2 to leave the disk alone or option 1 to add the disk to the list to be initialized in VxVM. If selecting option 1, a warning message is displayed about destroying data on the disk; however, **vxinstall** does not do any processing of the disk yet and there will be a chance to change this later in the process:

```
Are you sure (destroys data on hdisk1) [y,n,q,?] (default: n)
```

Type in **y** and then press Enter to go past this warning message. A prompt to rename the disk appears:

```
Enter disk name for hdisk1 [<name>,q,?] (default: disk01)
```

Type in the new name for the disk and then press Enter or just press Enter to accept the default name. A confirmation message is displayed that the disk has been selected. Press Enter to confirm initialization of the disk. At this point, the screen will go back to a similar one that was listed in Example 3-23, except it will list the second disk that has been detected. Go through the process for each disk that is detected. Once all disks have been selected and renamed as required, a screen similar to the one shown in Example 3-24 on page 62 will appear.

#### *Example 3-24 Custom installation confirmation screen*

---

Volume Manager Custom Installation  
Menu: VolumeManager/Install/Custom

The following is a summary of your choices.

|        |          |
|--------|----------|
| hdisk1 | New Disk |
| hdisk2 | New Disk |

Is this correct [y,n,q,?] (default: y)

---

If the correct disks have been selected, press Enter; otherwise, press n and then Enter. This will allow disks to be removed from the list to be initialized. Once all disks are selected correctly, press Enter, and **vxinstall** will now add and initialize the disks, set up the primary disk group rootdg, and then return to the command line. See Example 3-25 for an example of the process output.

#### *Example 3-25 Custom install initialization*

---

Is this correct [y,n,q,?] (default: y)

The Volume Manager is now reconfiguring (initializing phase)...

Volume Manager: Initializing hdisk1 as a new disk.

Volume Manager: Initializing hdisk2 as a new disk.

The Volume Manager is now reconfiguring (initialization phase)...

Volume Manager: Adding disk01 (hdisk1) as a new disk.

The Volume Daemon has been enabled for transactions.

Volume Manager: Adding disk02 (hdisk2) as a new disk.

---

### **Preventing multipathing or suppressing disks from VxVM**

“Prevent multipathing/Suppress devices from VxVM’s view” should be selected before selecting either the Quick Installation or Custom Installation options if needed. Once it is selected, a screen similar to the one shown in Example 3-26 appears.

#### *Example 3-26 Suppress multipathing selection*

---

Volume Manager Installation  
Menu: VolumeManager/Install/ExcludeDevices

This operation might lead to some devices being suppressed from VxVM's view or prevent them from being multipathed by vxdmp (This operation can be reversed using the vxdiskadm command).

Do you want to continue ? [y,n,q,?] (default: y)

---

As mentioned in Example 3-26 on page 62, any changes made here can be reversed later by using the **vxdiskadm** utility. See Chapter 4, “Basic administration” on page 77 for more details about the **vxdiskadm** command. Press Enter. The menu shown in Example 3-27 appears.

*Example 3-27 Suppress multipathing menu options*

---

Volume Manager Device Operations

Menu: VolumeManager/Install/ExcludeDevices

- 1 Suppress all paths through a controller from VxVM's view
- 2 Suppress a path from VxVM's view
- 3 Suppress disks from VxVM's view by specifying a VID:PID combination
- 4 Suppress all but one paths to a disk
- 5 Prevent multipathing of all disks on a controller by VxVM
- 6 Prevent multipathing of a disk by VxVM
- 7 Prevent multipathing of disks by specifying a VID:PID combination
- 8 List currently suppressed/non-multipathed devices
  
- ? Display help about menu
- ?? Display help about the menuing system
- q Exit from menus

Select an operation to perform:

---

Below is a brief description for each option. This menu has two sections. Options 1 to 4 are options to suppress paths from VxVM's view and are mainly used for disks attached to multiple controllers but are not supported by VxVM's multipathing feature. This means that if there are two or more paths to a disk and the disk array is not supported by VxVM's multipathing feature, then VxVM will see each path as a separate disk, and so options 1 to 4 need to be taken to explicitly exclude paths to a disk from VxVM's view. These disks will all be placed in the OTHER\_DISKS category and be treated as single path disks. Options 5 to 7 are options to stop VxVM from applying multipathing to selected disks if the disks are supported by VxVM's multipathing feature, and it will treat each selected disk as having only one path by selecting only one path to these disks. If any of options 1 to 7 are selected, then a reboot is required after installation. Option 8, “List currently suppressed/non-multipathed devices”, can be taken at any time to list what has been selected.

### ***Suppress all paths through a controller***

Use this option to specify a controller to exclude from the view of VxVM. This will exclude all paths via the selected controller to any disks that are attached to that controller from the view of VxVM. Multiple controllers can be entered. An example of a controller type is scsi1. Use the **lsdev** command to list controllers on the system. Example 3-28 shows the screen after option one is taken.

#### ***Example 3-28 Suppress all paths through a controller window***

---

Exclude controllers from VxVM

Menu: VolumeManager/Install/ExcludeDevices/CTLR-VXVM

Use this operation to exclude all paths through a controller from VxVM.

This operation can be reversed using the `vxdiskadm` command.

You can specify a controller name at the prompt. A controller name is of the form `scsi#` or `fscsi#`, example `scsi3`, `scsi11`, `fscsi2` etc. Enter 'all' to exclude all paths on all the controllers on the host. To see the list of controllers on the system, type 'list'.

Enter a controller name: [`<ctlr-name>`,all,list,list-exclude,q,?]

---

### ***Suppress a path from VxVM's view***

Using this option, you can specify a single path to a disk to suppress from VxVM's view. You can specify all paths, one path, or a list of paths to a disk. Example 3-29 shows the screen after taking option 2.

#### ***Example 3-29 Suppress a path from VxVM's view***

---

Exclude paths from VxVM

Menu: VolumeManager/Install/ExcludeDevices/PATH-VXVM

Use this operation to exclude one or more paths from VxVM.

As a result of this operation, the specified paths will be excluded from the view of VxVM. This operation can be reversed using the `vxdiskadm` command.

You can specify a pathname or a pattern at the prompt. Here are some path selection examples:

|         |                              |
|---------|------------------------------|
| all:    | all paths                    |
| hdisk6: | a single path                |
| list:   | list all paths on the system |

Enter a pathname or pattern : [`<pattern>`,all,list,list-exclude,q,?]

---



### ***Suppress disks from VxVM's view by specifying VID:PID***

Use this option to specify a Vendor ID and Product ID combination of disks to exclude. This will allow wildcards to be entered, for example, to exclude all Disks with a Vendor ID of IBM type in IBM:\*. Multiple entries can be placed on one line. To find the Vendor ID and Product ID of a disk, use the `/etc/vx/diag.d/vxdmping` command. For example, to find the Product ID and Vendor ID of `hdisk4`, run the following command:

```
# /etc/vx/diag.d/vxdmping /dev/hdisk4
Vendor id      : IBM
Product id     : DGHS09U
Revision       : 03E0
Serial Number  : 681DE948GAGS
```

Example 3-30 shows the screen after taking this option.

#### ***Example 3-30 Suppress disks from VxVM's view by specifying a VID:PID combination***

---

Exclude controllers from DMP

Exclude VID:PID from VxVM

Menu: VolumeManager/Install/ExcludeDevices/VIDPID-VXVM

Use this operation to exclude disks returning a specified VendorID:ProductID combination from VxVM.

As a result of this operation, all disks that return VendorID:ProductID matching the specified combination will be excluded from the view of VxVM. This operation can be reversed using the `vxdiskadm` command.

You can specify a VendorID:ProductID combination at the prompt. The specification can be as follows :

VID:PID                where VID stands for Vendor ID  
                         PID stands for Product ID  
                         (The command `vxdmping` in `/etc/vx/diag.d` can be  
                         used to obtain the Vendor ID and Product ID)

Both VID and PID can have an optional '\*' (asterisk) following them. If a '\*' follows VID, it will result in the exclusion of all disks returning Vendor ID starting with the specified VID. The same is true for Product ID as well. Both VID and PID should be non NULL. The maximum allowed lengths for Vendor ID and Product ID are 8 and 16 characters respectively.

Some examples of VID:PID specification are:

```
all                    - Exclude all disks
aaa:123                - Exclude all disks having VID 'aaa' and PID '123'
```

|          |  |
|----------|--|
| aaa*:123 | - Exclude all disks having VID starting with 'aaa' and PID '123' |
| aaa:123* | - Exclude all disks having VID 'aaa' and PID starting with '123' |
| aaa:*    | - Exclude all disks having VID 'aaa' and any PID                 |

Enter a VID:PID combination: [<pattern>,all,list-exclude,q,?]

---

### ***Suppress all but one path to a disk***

Use this option if a disk has multiple paths and you only want VxVM to use one, path to the disk. This option is used if the disk is not supported as part of VxVMs DMP feature. In this case, VxVM will see each disk path as a separate disk. To prevent this, all paths to a particular disk can be placed into a group called a pathgroup, which VxVM will recognize as one path to the disk. Enter all the paths to the disk and a pathgroup will be created allowing only one path to the disk via the pathgroup name. Example 3-31 shows the screen after this option has been taken.

#### ***Example 3-31 Exclude all but one path to a disk***

---

Exclude all but one paths to a disk

Menu: VolumeManager/Install/ExcludeDevices/PATHGROUP-VXVM

Use this operation to exclude all but one paths to a disk. In case of disks which are not multipathed by vxdmp, VxVM will see each path as a disk. In such cases, creating a pathgroup of all paths to the disk will ensure that only one of the paths from the group is made visible to VxVM. The pathgroup can be removed using the vxdiskadm command.

Example: If hdisk6 and hdisk16 are paths to the same disk and both are seen by VxVM as separate disks, hdisk6 and hdisk16 can be put in a pathgroup so that only one of these paths is visible to VxVM.

The pathgroup can be specified as a list of blank separated paths, for example, hdisk6 hdisk16.

Enter pathgroup: [<pattern>,list,list-exclude,q,?]

---

### ***Prevent multipathing of disks on a controller***

Use this option to prevent VxVM from applying multipathing to disks on a controller. Specify the controller to prevent multipathing of disks on. This will place the disks on this controller in the OTHER\_DISKS category and ignore all other paths to the disks. Example 3-32 on page 67 shows the screen after this option has been taken.

### *Example 3-32 Prevent multipathing of disks on a controller*

---

Exclude controllers from DMP

Menu: VolumeManager/Install/ExcludeDevices/CTLR-DMP

Use this operation to exclude all disks on a controller from being multipathed by vxddmp.

As a result of this operation, all disks having a path through the specified controller will be claimed in the OTHER\_DISKS category and hence, not multipathed by vxddmp. This operation can be reversed using the vxddiskadm command.

You can specify a controller name at the prompt. A controller name is of the form scsi# or fscsi#, example scsi3, scsi11, fscsi2 etc. Enter 'all' to exclude all paths on all the controllers on the host. To see the list of controllers on the system, type 'list'.

Enter a controller name: [<ctlr-name>,all,list,list-exclude,q,?]

---

### ***Prevent multipathing of a disk by VxVM***

Use this option for disks that have multiple paths but only one path to the disk is needed. Example 3-33 shows the screen after this option has been selected. Type in the disk to prevent multipathing on and then press Enter.

### *Example 3-33 Prevent multipathing of a disk by VxVM*

---

Exclude paths from DMP

Menu: VolumeManager/Install/ExcludeDevices/PATH-DMP

Use this operation to exclude one or more disks from vxddmp.

As a result of this operation, the disks corresponding to the specified paths will not be multipathed by vxddmp. This operation can be reversed using the vxddiskadm command.

You can specify a pathname or a pattern at the prompt. Here are some path selection examples:

|         |                              |
|---------|------------------------------|
| all:    | all paths                    |
| hdisk6: | a single path                |
| list:   | list all paths on the system |

Enter a pathname or pattern : [<pattern>,all,list,list-exclude,q,?]

---

### ***Prevent multipathing of disks by specifying a VID:PID combination***

Use this option to specify a Vendor ID or Product ID of disks to be excluded from multipathing. For each Vendor ID or Product ID excluded, the disk will be placed in the OTHER\_DISKS group. Refer to “Suppress disks from VxVM’s view by specifying VID:PID” on page 65 for information on finding the VID:PID combination using the **vxddmpinq** command. Example 3-34 shows the screen after this option has been selected.

#### ***Example 3-34 Prevent multipathing of disks by specifying a VID:PID combination***

---

Exclude VID:PID from DMP

Menu: VolumeManager/Install/ExcludeDevices/VIDPID-DMP

Use this operation to prevent vxddmp from multipathing disks returning a specified VendorID:ProductID combination.

As a result of this operation, all disks that return VendorID:ProductID matching the specified combination will be claimed in the OTHER\_DISKS category (ie, they will not be multipathed by vxddmp). This operation can be reversed using the vxddiskadm command.

You can specify a VendorID:ProductID combination at the prompt. The specification can be as follows :

VID:PID            where VID stands for Vendor ID  
                     PID stands for Product ID  
                     (The command vxddmpinq in /etc/vx/diag.d can be  
                     used to obtain the Vendor ID and Product ID)

Both VID and PID can have an optional '\*' (asterisk) following them. If a '\*' follows VID, it will result in the exclusion of all disks returning Vendor ID starting with the specified VID. The same is true for Product ID as well. Both VID and PID should be non NULL. The maximum allowed lengths for Vendor ID and Product ID are 8 and 16 characters respectively.

Some examples of VID:PID specification are:

|          |  |
|----------|--|
| all      | - Exclude all disks  |
| aaa:123  | - Exclude all disks having VID 'aaa' and PID '123'               |
| aaa*:123 | - Exclude all disks having VID starting with 'aaa' and PID '123' |
| aaa:123* | - Exclude all disks having VID 'aaa' and PID starting with '123' |
| aaa:*    | - Exclude all disks having VID 'aaa' and any PID                 |

Enter a VID:PID combination: [<pattern>,all,list-exclude,q,?]

---

### 3.3.3 Post-installation verification

In this section, we describe some verification and other related tasks after installation.

#### VERITAS Volume Manager daemons

Once **vxinstall** has been run, the next step is to check the VxVM **daemons** have started. This can be done by running the following command:

```
# ps -ef | grep vx
```

This should show one **vxconfigd** process running as follows:

```
root 573636      1   0 15:53:21      -   0:00 vxconfigd -k -m enable
```

If the system was rebooted after running **vxinstall**, you may also see the following processes:

```
# ps -ef | grep -i vx
root 19374 22450   0   Jul 26      -   0:00 vxnotify -f -w 15
root 22450      1   0   Jul 26      -   0:00 sh - /usr/lib/vxvm/bin/vxrelocd
root 38368      1   0   Jul 31      -   0:06 vxconfigd -k -m disable
root 38928 39318   1 14:29:25 pts/1   0:00 grep -i vx
root 40052      1   0   Jul 29      -   0:51 /opt/VRTSob/bin/vxsvc
```

**vxnotify** and **vxrelocd** are started as part of the system startup scripts. This will be covered in Chapter 4, “Basic administration” on page 77.

If the **vxconfigd** process is not running, it can be started by running the following two commands:

```
# /usr/sbin/vxconfigd -k
# /usr/sbin/vxdctl enable
```

Also, the following command should be run to check the **vxiod** processes. The **vxiod** process is a kernel process and cannot be seen using the **ps** command:

```
# vxiod
```

This should return the following output:

```
10 volume I/O daemons running
```

If there are no **vxiod** processes running, run the following command:

```
# vxiod set 10
```

## Startup scripts

The VxVM installation process will place the following system startup scripts that are called during a system reboot in the `/etc/rc.d/rc2.d` directory:

- ▶ S50isid
- ▶ S94vxnm-host\_infod
- ▶ S94vxnm-vxnetd
- ▶ S95vxvm-recover
- ▶ S96vradmind
- ▶ S96vxrsyncd

These scripts are symbolic links to scripts in `/etc/init.d` and are called from the following line in the `inittab`:

```
12:2:wait:/etc/rc.d/rc 2
```

After the above scripts are run, the following scripts are run from the `/etc/inittab`:

```
voll::bootwait:/etc/init.d/vxvm-sysboot >/dev/console 2>&1  
vol3::bootwait:/etc/init.d/vxvm-startup2 >/dev/console 2>&1  
vol4:2:wait:/etc/init.d/vxvm-recover >/dev/console 2>&1
```

These scripts will start up the volume manager daemons and start all volumes as well as starting the VEA back-end server. For a description of these startup scripts, see Chapter 4, “Basic administration” on page 77.

## Man pages and PATH variable

To access the VxFS and VxVM Man pages, set up the following environment variable:

```
MANPATH=$MANPATH:/opt/VRTS/man ; export MANPATH
```

or for C shell users:

```
setenv MANPATH $MANPATH:/opt/VRTS/man
```

To access the VxFS and VxVM commands, ensure the following paths are in the `PATH` environment variable: `/sbin`, `/usr/sbin`, `/opt/VRTS/bin` and `/etc/vx/bin`.

These can be added to a users `.profile`, `.kshrc`, or the `/etc/profile` for automatic setup at login time or to the `.cshrc` file for C shell users.

## Starting the VEA server

To start the VERITAS Enterprise Administrator (VEA) back-end server, run the following command:

```
# /opt/VRTSob/bin/vxsvc
```

This will also get started during system startup by the `/etc/rc.d/rc2.d/S50isisd` script.

To test the Visual Administrator is working, run the following command:

```
# /opt/VRTSob/bin/vea &
```

This will start the GUI interface. For more information on using the VEA GUI, see Chapter 4, “Basic administration” on page 77.

**Note:** If there are problems starting the VEA GUI, check the `smit.log` for the installation of VRTSobgui. At the time of the writing of this redbook, the **installp** process puts a tar file in `/opt/VRTSob` called `jre.tar` and then extracts it. If there is not enough file system space available, this extract will fail, but the status of the install of VRTSobgui will be listed as successful in the `smit.log`. To check if this has happened, look for the following entry in the `smit.log`:

```
installp: APPLYING software for:
          VRTSobgui 3.0.255.1

restore: 0511-138 Cannot write to file
./usr/lpp/VRTSobgui/inst_root/opt/VRTSob/
jre.tar.
Restore : There is not enough space in the file system.
0503-700 inurest: Error in restoring files
```

If this happens, increase the file system and use the **tar** command to extract the `/opt/VRTSob/jre.tar` file. If the problem persists, re-install VRTSobgui.

## Installing VRTSexplorer

VRTSexplorer is an optional package that does not require an additional license. It is used for gathering information for VERITAS support in the event of problems that you may need to log a call to VERITAS about. VRTSexplorer is located on the VERITAS Foundation Suite for AIX installation CD and on the AIX 5L bonus pack CD. Alternatively it can be downloaded from:

<ftp://ftp.veritas.com/pub/support/vxexplore.tar.Z>

To install VRTSexplorer, change the directory to `/cdrom/support` if using the VERITAS Foundation Suite for AIX Installation CD or to `/cdrom/other/VRTSFST/support` if installing from the AIX 5L bonus pack CD and run the following command:

```
# installp -acXd /cdrom/support/VRTSspt.bff VRTSspt
```

### 3.3.4 Uninstalling VxFS and VxVM

Before removing VxFS and VxVM, make sure all data is moved off any file systems that are mounted on VERITAS Volumes if it is required to be kept. If there is not enough disk space available in the LVM's Volume Group, then disks may be able to be evacuated from VxVM to free up disks and re-added to the LVM's Volume Group. Evacuation is a procedure that moves data from one disk to another and in some cases can be used for consolidation. For more information on evacuating subdisks, refer to 5.5.3, "Evacuating volumes from a disk" on page 169.

Once the Logical Volume has been created and a new file system has been created, data can be moved from the VERITAS file system to the newly created file system by any one of the following methods:

- ▶ Backing up the data on the VERITAS file system and restoring it to the newly created file system.
- ▶ Using the **dd** command to copy the device file to the new device file that the file system is mounted on (see Example 3-35).
- ▶ Using **tar** or **cpio** to copy across the contents of the VERITAS file system to the newly created file system (see Example 3-36).

*Example 3-35 Using dd to copy contents of VERITAS file system to LVM file system*

---

```
dd if=/dev/vx/dsk/datadg/VRTSvolume of=/dev/LVMvolume
```

---

*Example 3-36 Using tar to copy contents of VERITAS file system to LVM file system*

---

```
ksh
cd /VRTSfilesystem
tar -cvpf - . | (cd /LVMfilesystem ; [[ $? -eq 0 ]] && tar -xvpf - )
```

---

Once all VERITAS file systems have been moved onto LVM file systems, unmount all VERITAS file systems. Stop and delete all VERITAS volumes running on the system; this can be done as follows:

1. Find a list of all volumes on the system by running the following script:

```
# for DG in `vxdg list | awk 'NR>1 {print $1}'`
> do
> vxinfo -g $DG | awk '{print $1}'
> done
```

2. For each volume returned by the above script, run the following commands:

```
# vxvol stop <volume>
# vxedit -rf rm <volume>
```

3. List all disk groups on the system by running the following command:

```
# vxdg list
```



4. Remove all disk groups except the rootdg by running the following command for each disk group:

```
# vxdg destroy <diskgroup>
```

5. List all disks left in the rootdg by running the following command:

```
# vxdisk -g rootdg list
```

6. Remove all disks from the rootdg except for one, as you cannot remove the last disk from the rootdg. Use the following command to do this for each disk:

```
# vxdg rmdisk <diskname>
```

7. List all disks in use by VxVM with the following command:

```
# lspv
hdisk0      000321941dc75aeb      VeritasVolumes
hdisk1      00032194faa00f1f      rootvg
hdisk2      00032194430983ee      VeritasVolumes
hdisk3      000321944957d841      VeritasVolumes
hdisk4      000321946f05b50d      None
```

8. For each disk, apart from the one left in the rootdg, which has a status of VeritasVolume in the right hand column, run the following command:

```
# /etc/vx/bin/vxdiskunsetup -C <disk>
```

This will remove the status of Veritasvolume when running the **lspv** command and allow the disk to be re used by the LVM.

9. Stop the VEA back-end server by running the following command:

```
# /opt/VRTSob/bin/vxsvc -k
```

10. Shut down the volume manager by running the following commands:

```
# vxdctl stop
# vxiod -f set 1
```

11. Uninstall the VERITAS Foundation Suite for AIX packages by using either SMIT or **installp**. To uninstall using SMIT, see “Uninstalling using SMIT” on page 73. To uninstall using **installp**, see “Uninstalling using installp command” on page 75.

## Uninstalling using SMIT

To uninstall using SMIT, the following smitty fast path can be used:

```
# smitty remove
```

The screen shown in Example 3-37 on page 74 appears.

Example 3-37 SMIT uninstall screen

Remove Installed Software

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

\* SOFTWARE name

PREVIEW only? (remove operation will NOT occur)

REMOVE dependent software?

EXTEND file systems if space needed?

DETAILED output?

[ ]

yes

no

no

no

+

+

+

+

+

F1=Help

F2=Refresh

F3=Cancel

F4=List

F5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do

With the cursor in the software name field, press F4. A list of software appears. Press the “/” key. Type VRTS into the search pattern. This now goes to the top of the list of packages starting with VRTS. Select each package by pressing F7 with the cursor next to the package name. Use the arrow keys to move up and down. Select the packages shown in Example 3-38.

Example 3-38 SMIT remove software selection window

Remove Installed Software

SOFTWARE name

Move cursor to desired item and press F7.

ONE OR MORE items can be selected.

Press Enter AFTER making all selections.

[MORE...13]

> VRTSfsdoc

> VRTSfspro

> VRTSob

> VRTSobgui

> VRTSvlic

> VRTSvmdoc

> VRTSvmman

> VRTSvmpo

> VRTSvxfs

[MORE...299]

F1=Help

F2=Refresh

F3=Cancel

|                     |          |             |  |
|---------------------|----------|-------------|--|
| F7=Select           | F8=Image | F10=Exit    |  |
| Enter=Do            | /=Find   | n=Find Next |  |
| +-----+-----+-----+ |          |             |  |

---

Once all packages are selected, press Enter. The “Remove Software” main window appears. Arrow down to the “PREVIEW only? (remove operation will NOT occur)” field and press the Tab key; this will change to no. Arrow down to “REMOVE dependent software?” and press the Tab key to change this to yes. Press Enter, and a warning message appears. Press Enter again, and the uninstall will start. A log file of the uninstall will be kept in the smit.log located in the root users home directory.

Once the filesets are uninstalled, proceed to “Moving disks back to control of LVM” on page 75.

## Uninstalling using installp command

To uninstall using **installp**, run the following command:

```
# installp -u VRTSfsdoc VRTSvxfs VRTSvmman VRTSvmdoc VRTSob VRTSobgui VRTSfspro
VRTSvmpo VRTSvxvm VRTSvlic
```

Once the filesets are uninstalled, proceed to “Moving disks back to control of LVM” on page 75.

## Moving disks back to control of LVM

Once all VxFS and VxVM packages are uninstalled, the last disk that was still in the rootdg disk group will have a status of VeritasVolumes when running the **lspv** command. Run the **lspv** command to see if there are any disks with a status of VeritasVolumes as follows:

```
# lspv
hdisk0          000bc6cdc3e0502a          rootvg
hdisk1          000bc6cdf35adc98          VeritasVolumes
hdisk2          000bc6cddec6d4d5          rootvg
hdisk3          000bc6cddec73d1b          rootvg
hdisk4          000bc6cd71b06a6f          rootvg
```

For each disk of this type, run the following command:

```
# chpv -C <disk>
```

These disks can now be added to an LVM Volume Group.



# Basic administration

In this chapter, we describe the procedures to carry out the common administration tasks and the methods that are available to perform these tasks. The following topics are discussed:

- ▶ System startup and process control
- ▶ Methods of administration
  - Command line interface (CLI)
  - VERITAS Enterprise Administrator Java GUI (VEA)
  - VERITAS supplied menu administration utilities
  - SMIT
- ▶ Basic administration tasks
  - Adding disks
  - Creating disk groups
  - Creating volumes
  - Creating file systems
  - Resizing volumes and file systems
  - Failure monitoring

Also, the following terminology associated with VERITAS File System (VxFS) and VERITAS Volume Manager (VxVM) will be used:

- ▶ Disks
- ▶ Subdisks
- ▶ Plexes
- ▶ RAID
- ▶ Mirroring

For definitions of these terms, refer to Chapter 2, “Components” on page 9.

## 4.1 System startup and process control

This section provides an overview of the startup process and the scripts that are called during the system boot process. It also describes the VxVM processes and the commands used to control them.

### 4.1.1 Startup process

This section provides a description of the startup process during a system boot. It includes descriptions of the startup scripts found in the `/etc/init.d` directory, the processes they start, and the commands used to control these processes.

The VxVM processes are started when the system enters run level two and the following scripts found in the `/etc/inittab` are run:

```
vol1::bootwait:/etc/init.d/vxvm-sysboot >/dev/console 2>&1
vol3::bootwait:/etc/init.d/vxvm-startup2 >/dev/console 2>&1
vol4:2:wait:/etc/init.d/vxvm-recover >/dev/console 2>&1
l2:2:wait:/etc/rc.d/rc 2
```

After the first three scripts are run, the `/etc/rc.d/rc` script is called with a parameter of 2. This script will look for all scripts in the `/etc/rc.d/rc2.d` directory that start with an upper case S, such as `S94vxnm-recover`, and an upper case K, such as `K50isisd`. Each script that it finds with an upper case S will be run with a “start” parameter and scripts in this directory that start with an upper case K are run with a “stop” parameter. The number after the S or K in the name of these scripts determines the order in which the scripts are run, lower numbered scripts are run before higher numbered scripts, and scripts starting with K are run before scripts starting with S. The scripts that VxVM places in this directory during installation are as follows:

- ▶ `K50isisd`
- ▶ `S50isisd`
- ▶ `S94vxnm-host_infod`
- ▶ `S94vxnm-vxnetd`
- ▶ `S95vxvm-recover`
- ▶ `S96vradmind`
- ▶ `S96vxrsyncd`

**Note:** At time of writing, the install process places `K50isisd` in the `/etc/rc.d/rc2.d` directory. This script is intended for shutdown, but actually gets run on startup. To fix this, move the script to `/etc/init.d` and call it from `rc.shutdown`. Also, the `S95vxvm-recover` script which is a symbolic link to `/etc/init.d/vxvm-recover`, is run twice: once from the `inittab` and once from the `/etc/rc.d/rc2.d` directory. The symbolic link can be removed from the `/etc/rc.d/rc2.d` directory.

Some of these scripts are symbolic links to files in the `/etc/init.d/` directory. The following sections describes the purpose of each startup script.

### **vxvm-sysboot**

This script starts up the **vxconfigd** daemon, which is required to perform VxVM updates on objects such as disk groups and volumes. This script also determines whether the root (`/`) and `/usr` file systems are encapsulated and, if so, whether the system can be booted off the encapsulated volumes. As encapsulation of the root file system is not yet possible under AIX, this section of the script is not relevant.

### **vxvm-startup2**

This script checks that **vxconfigd** is already running and, if not, starts it. It then starts the **vxiod** kernel, processes and imports all disks groups, and starts all volumes.

### **vxvm-recover**

This script attaches and synchronizes any plexes on volumes as needed. It also starts up the **vxrelocd** daemon, which is used for hot relocation. Alternatively, this script can start up hot sparing via the **vxsparecheck** daemon instead of hot relocation by commenting out the following line as follows:

```
#vxrelocd root &
```

And by uncommenting the following line:

```
vxsparecheck root &
```

For more information on hot relocation, see 5.5.1, “Hot relocation” on page 167, and for more information on hot sparing, see 5.5.2, “Hot sparing” on page 168.

### **K50isisd**

This script is used to shut down the VEA back-end server that runs as process **vxsvc**.

### **S50isisd**

This script starts up the VEA back-end server by starting the **vxsvc** process.

### **S94vxnm-host\_infod, S94vxnm-vxnetd, S96vradmind and S96vxrsyncd**

These four scripts are only used if VERITAS Volume Replicator (VVR) is installed and licensed on the system. The scripts are called as part of the startup process but only do processing if a VVR license is detected. VVR is not part of the



VERITAS Foundation Suite; however, these scripts and other utilities are installed with VxVM in preparation for use with VVR.

**Note:** At the time of the writing of this redbook, there are no shutdown scripts for VxVM. This does not cause any problems; however, to stop VxVM cleanly before system shutdown, a script can be put in place that can be called from rc.shutdown that contains the following lines:

```
/usr/sbin/vxiod -f set 0  
/usr/sbin/vxdctl stop
```

## 4.1.2 Managing processes

After a reboot, the following processes are usually started up:

- ▶ vxconfigd
- ▶ vxiod kernel processes
- ▶ vxrelocd or vxsparecheck
- ▶ vxnotify
- ▶ vxsvc

The first two processes are required for VxVM to function correctly and the other processes are processes to help with administration and do not need to be running in order to use VxVM. The following sections describe these processes and options for changing the mode of them.

### vxconfigd

This process is known as the VxVM configuration daemon and is responsible for maintaining the disk group and volume configuration. It reads information from the `/etc/vx/volboot` file, which contains information about the host name and the disks in the rootdg disk group. There are two main modes for this process: enabled and disabled. In order to perform most VxVM operations, it needs to be in the enabled mode. In disabled mode, vxconfigd will not allow any VxVM configuration changes or listing of VxVM objects; however, applications can still use file systems mounted on VxVM volumes as normal. To determine the mode vxconfigd is in, use the **vxdctl** command as follows:

```
# vxdctl mode  
mode: enabled
```

The mode of **vxconfigd** can also be managed via the **vxdctl** command. In order to set the vxconfigd mode to disabled, run the following command:

```
# vxdctl disable
```

To set the mode to enabled, run the following command:

```
# vxdctl enable
```

After system startup, the **ps** command will show the **vxconfigd** running as follows:

```
# ps -ef | grep vxconfigd
root 6738      1  0 14:36:22      -  0:04 vxconfigd -m boot
```

The **-m** option of **vxconfigd** will show the initial mode, but the **vxdctl** command should be used to determine the current mode.

## **vxiod**

The **vxiod** command is used to report on the number of VxVM kernel I/O processes running and to start and stop these processes. These processes will not show up in the output of the **ps** command, as they are kernel processes. In order to determine whether these processes are running, run the following command:

```
# vxiod
10 volume I/O daemons running
```

The number of **vxiod** processes can be changed as required by using the **vxiod** command. The maximum number of allowed processes is 64 and the minimum is 1; if more than 64 is specified with the **vxiod** command, it will only set 64. To change the number of processes, run the commands as follows:

```
# vxiod set 15
# vxiod
15 volume I/O daemons running
```

## **vxrelocd and vxsparecheck**

These two processes are used by VxVM in the event of disk failures for hot sparing or hot relocation. Both processes cannot be running at the same time. One of these processes is started from the **/etc/init.d/vxvm-recover** script at system startup; by default, it is **vxrelocd**, but this can be changed by editing the script. Alternatively, both can be disabled. For more information on hot relocation, see 5.5.1, “Hot relocation” on page 167, and for more information on hot sparing, see 5.5.2, “Hot sparing” on page 168.

## **vxnotify**

This process is also started from the **/etc/init.d/vxvm-recover** script at system startup time with the **-f** option; its purpose is to monitor for VxVM object changes, such as disk and plex failures. The **-f** option will monitor for disk, volume, and plex detaches; if any of these are found, it will send a message to root’s mail by

default via either **vxrelocd** or **vxsparecheck**, which monitors the output from **vxnotify**.

### **VXSVC**

This process runs the VERITAS Enterprise Administrator (VEA) back-end server, which enables connections from multiple clients, allowing remote administration via VEA. To determine whether the back-end server is running, run the following command:

```
# /opt/VRTSob/bin/vxsvc -m  
Current state of server : RUNNING
```

To stop the back-end server, run the following command:

```
# /opt/VRTSob/bin/vxsvc -k  
Request is in process..  
Server is shutting down .  
Server was shutdown successfully
```

## **4.2 Methods of administration**

In this section, we discuss the different ways it is possible to perform administration tasks. With VxVM, there are many different ways to achieve the same result. The methods discussed in this section are:

- ▶ Command line interface (CLI)
- ▶ VEA java GUI
- ▶ VERITAS supplied utilities
- ▶ Using SMIT

### **4.2.1 Command-line interface**

VxVM provides a lot of commands that can be used from the command line to perform VxVM operations, such as creating disk groups and volumes and commands to display information. Examples of these commands are **vx dg**, **vxvol**, and **vxprint**. These commands are usually located in the **/sbin**, **/usr/sbin**, and **/etc/vx/bin** directories.

## 4.2.2 VEA Java GUI

VxVM provides a Java based Graphical User Interface (GUI) for performing administration tasks. In order to use this, the back-end server must be running on the host. If the server is not running, it can be started by running the following command:

```
# /opt/VRTSob/bin/vxsvc
```

The VEA can be used to administer remote systems if the client software is installed on the local system.

The VEA client can also be loaded onto a PC running the Microsoft Windows operating system. To install the VEA client on a windows PC, the VRTSobgui.msi can be obtained from the VERITAS Foundation Suite installation CD or the AIX 5L Version 5.1 bonus pack CD in the /cdrom/other/VRTSFST/win32 directory. This file needs to be installed via the Microsoft Installer. Once installed, it can be accessed by clicking on **Start -> Programs -> Veritas -> VERITAS Enterprise Administrator**.

To start the VEA GUI on a UNIX host, ensure the DISPLAY environment variable is set correctly and run the following command:

```
# /opt/VRTSob/bin/vea &
```

The VEA login screen will appear, as shown in Figure 4-1 on page 85.

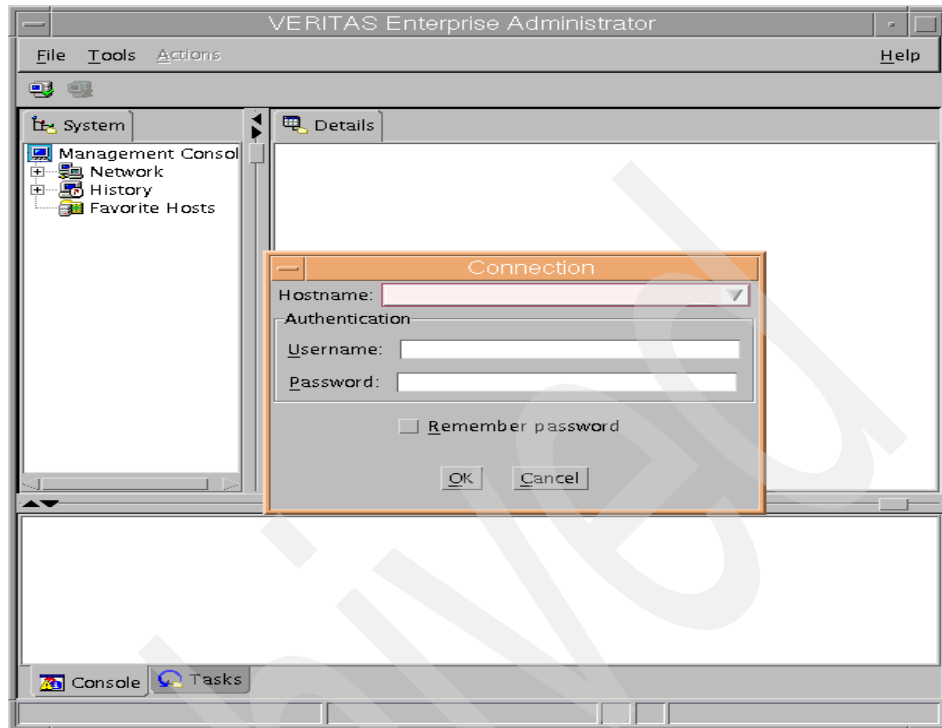


Figure 4-1 VEA login screen

To log in to the VEA, click on the **Hostname** tab in the login box; this will bring down a list of hosts that the VEA client has detected on the network. A host can be selected by clicking the selected host name. If there are no host names listed in the drop down window, click **Cancel** in the login window, and then right-click on the **Network** icon that appears underneath the **Management Console** icon. A **Refresh** icon will appear; click on **Refresh**, which will start a search across the network to find any hosts that have a back-end server running. When the list is updated, select **File -> Connect** at the top of the window; the login prompt will reappear. Select the host to connect to from the drop-down box; this will put the user "root" ID in the **Username** field. Type in the password in the password field and then click OK.

To use a user ID other than root, the user ID needs to be in a group called `vrtsadm`. If this group does not exist, it needs to be created. Alternatively, the administration group name can be changed from `vrtsadm` to another name by editing the following line in the `/opt/VRTSob/config/Registry` file:

```
[REG_SZ] "AccessGroups" = "vrtsadm";
```

Change the name from vrtsadm to the new name, then stop and start the back-end server using the following commands:

```
# /opt/VRTSob/bin/vxsvc -k
Request is in process..
Server is shutting down .
Server was shutdown successfully
# /opt/VRTSob/bin/vxsvc
# /opt/VRTSob/bin/vxsvc -m
Current state of server : RUNNING
```

Once logged on, the window appears as shown in Figure 4-2.

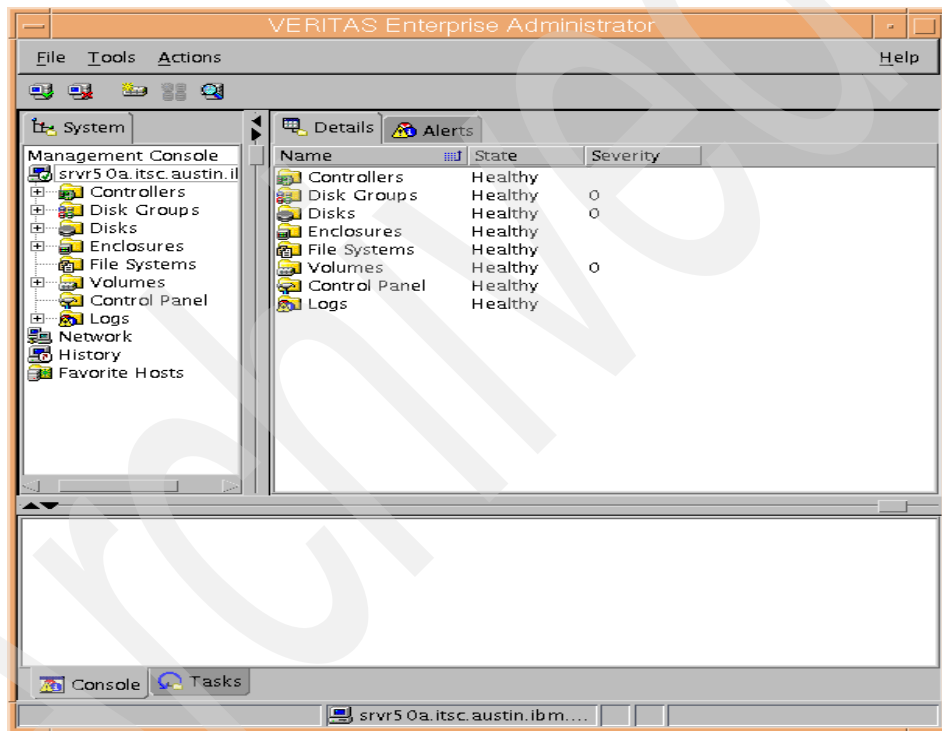


Figure 4-2 VEA main window

Once logged in, the VEA is divided into three subwindows, as shown in Figure 4-2. They are as follows:

- ▶ Object tree and grid on the left hand side.
- ▶ Status area at the bottom of the screen.
- ▶ The grid window on the right hand side, which displays whatever is selected on the left hand side.

When signed into a host, the **Host** icon can be double-clicked in the left hand side subpanel and the VxVM objects will be listed beneath it, such as controllers, disk groups, disks, enclosures, file systems and volumes. Clicking on any of these will show the details in the right hand side window. The status area at the bottom of the screen will show any error events for VxVM objects. Clicking on the **Tasks** tab below the status subpanel will show a list of administration tasks that have been performed during the session. The command that the VEA uses to perform the task can be found by right-clicking on it. A drop-down menu appears; select **Properties** from the menu. The command used to perform the task is then displayed. This is a useful way to learn how to use the CLI to perform administration tasks.

The menu bar across the top provides four options:

- ▶ **File:** Can be used for signing into new hosts or disconnecting from current hosts. Provides object properties, depending on what is active in the right-hand side window. Can be used to exit VEA.
- ▶ **Tools:** Used for changing user preferences.
- ▶ **Actions:** The options in this menu vary, depending on what object is selected and is active in the right-hand side subwindow.
- ▶ **Help:** Provides online help.

**Note:** The VEA uses different terminology for some VxVM objects. A disk group is known as a Dynamic disk group and a mirrored-striped (RAID-10) is known as a striped pro.

### 4.2.3 VERITAS supplied utilities

For some VxVM administration tasks, there are menu-based utilities available that can be run from the command line. An example of this is the **vxdiskadm** utility. When these utilities are run from the command line, they will appear with a list of options that can be selected by typing in the number to the left of the option and pressing Enter.

### 4.2.4 Using SMIT

SMIT can be used to perform VxVM and VxFS administration tasks.

To perform VxFS tasks using SMIT, select **System Storage Management (Physical & Logical Storage) -> File Systems -> Add/Change/Show/Delete File Systems -> VERITAS File System (VxFS)**, or the **smitty vxfs** fast path can be used. Example 4-1 on page 88 shows the menu options available for administering VxFS under SMIT.

Example 4-1 SMIT VxFS menu

| VERITAS File System (VxFS)                               |            |           |          |
|--|------------|-----------|----------|
| Move cursor to desired item and press Enter.             |            |           |          |
| Add a VERITAS File System on a Previously Defined Volume |            |           |          |
| Mount a VERITAS File System                              |            |           |          |
| Change / Show Characteristics of a VERITAS File System   |            |           |          |
| Remove a VERITAS File System                             |            |           |          |
| F1=Help  | F2=Refresh | F3=Cancel | F8=Image |
| F9=Shell   | F10=Exit   | Enter=Do  |          |

To perform VxVM tasks using SMIT, select **System Storage Management (Physical & Logical Storage)** -> **VERITAS Volume Manager**, or the **smitty vxvm** fast path can be used. Example 4-2 shows an example of the options available for administering VxVM under SMIT.

Example 4-2 SMIT VxVM menu

| VERITAS Volume Manager                       |            |           |          |
|--|------------|-----------|----------|
| Move cursor to desired item and press Enter. |            |           |          |
| Disk Groups                                  |            |           |          |
| VxVM Volumes                                 |            |           |          |
| VxVM Disk Administrator                      |            |           |          |
| Change / Show VxVM Tunables                  |            |           |          |
| Configure Replicated Data Set (RDS)          |            |           |          |
| Replication Tasks                            |            |           |          |
| F1=Help                                      | F2=Refresh | F3=Cancel | F8=Image |
| F9=Shell                                     | F10=Exit   | Enter=Do  |          |

4.3 Basic administration tasks

In this section, we describe how to perform the basic day-to-day administration tasks associated with VxVM and VxFS. For each task, there can be multiple ways of achieving the same result. This section will endeavor to list all the possible options. For first time users of VERITAS Foundation Suite, the VEA is possibly the best option for administration; as users become more familiar with VxVM and VxFS, the CLI is more useful in achieving tasks more efficiently.



For more information on the commands or processes mentioned in this section, refer to the following publications:

- ▶ *VERITAS File System 3.4 - Administrators Guide (AIX)*
- ▶ *VERITAS Volume Manager 3.2 - Administrators Guide (AIX)*
- ▶ *VERITAS Volume Manager 3.2 - Users Guide (AIX)*
- ▶ Online man pages

### 4.3.1 Adding disks

Disks can be added to VxVM either by directly adding them to a disk group or by leaving them out of a disk group for use later. To see how to add a disk directly to a disk group, see 4.3.2, “Creating disk groups” on page 93. Disks that are under control of the AIX Logical Volume Manager cannot be initialized for use under VxVM. Use the **chpv -C** command to remove a disk from LVM control. When a disk is added to VxVM, it places its own configuration information in an area known as the private region at the start of the disk.

#### Adding disks using the CLI

To set up a disk for use with VxVM, use the **vxdisksetup** command. To initialize disk **hdisk7** for use with VxVM, run the following command:

```
# vxdisksetup -i hdisk7
```

The **vxdisk** command can also be used to initialize a disk for use with VxVM. Example 4-3 shows an example of adding a disk to VxVM using **vxdisk**. The **lspv** command is run before and after the initialization. Note the status has changed from “None” to “VeritasVolumes” after the initialization. The **-f** option can be used with **vxdisk** to force the initialization; this may be needed if the following message appears and you are sure the disk is not in use by LVM:

```
# vxdisk init hdisk4
vxvm:vxdisk: ERROR: Device hdisk4: define failed:
    Disk already initialized
```

*Example 4-3 Adding disks to VxVM using vxdisk*

---

```
# lspv
hdisk0      000321941dc75aeb      VeritasVolumes
hdisk1      00032194faa00f1f      rootvg
hdisk2      00032194430983ee      VeritasVolumes
hdisk3      000321944957d841      VeritasVolumes
hdisk4      000321946f05b50d      None

# vxdisk init hdisk4

# lspv
hdisk0      000321941dc75aeb      VeritasVolumes
```

|        |                  |                |
|--------|------------------|----------------|
| hdisk1 | 00032194faa00f1f | rootvg         |
| hdisk2 | 00032194430983ee | VeritasVolumes |
| hdisk3 | 000321944957d841 | VeritasVolumes |
| hdisk4 | 000321946f05b50d | VeritasVolumes |

---

Another command that can be used to add disks to VxVM is **vxdiskadd**. This command will prompt for a series of responses. To add a disk, run **vxdiskadd** from the command line, followed by the disk to add as a parameter, for example:

```
# vxdiskadd hdisk4
```

A confirmation message will appear; press Enter. A message appears about the different options available to add the disk followed by the following prompt:

```
Which disk group [<group>,none,list,q,?] (default: rootdg)
```

Type in none and then press Enter; a message that the disk will be left free and not added to a disk group appears, followed by the following message:

```
Continue with operation? [y,n,q,?] (default: y)
```

Press Enter; the following message appears:

```
Use a default private region length for the disk?
[y,n,q,?] (default: y)
```

Press Enter. The disk is then initialized for use with VxVM.

## Adding disks using the VEA

To add a disk to VxVM using VEA, it also needs to be added to a disk group in the same process (disks cannot be added to VxVM without going into a disk group via VEA). To find out how to add a disk to a disk group via VEA, see “Creating disk groups using VEA” on page 93.

## Adding disks using vxdiskadm

The **vxdiskadm** utility can be used for adding disks to VxVM by either placing them into a disk group or leaving them out of a disk group for later use. **vxdiskadm** will prompt for similar user responses as those found by using the **vxdiskadd** command. To add a disk to VxVM, run the **vxdiskadm** command. The screen shown in Example 4-4 appears.

### Example 4-4 vxdiskadm main menu

---

```
Volume Manager Support Operations
Menu: VolumeManager/Disk
```

- 1 Add or initialize one or more disks
- 2 Remove a disk

```

3      Remove a disk for replacement
4      Replace a failed or removed disk
5      Mirror volumes on a disk
6      Move volumes from a disk
7      Enable access to (import) a disk group
8      Remove access to (deport) a disk group
9      Enable (online) a disk device
10     Disable (offline) a disk device
11     Mark a disk as a spare for a disk group
12     Turn off the spare flag on a disk
13     Unrelocate subdisks back to a disk
14     Exclude a disk from hot-relocation use
15     Make a disk available for hot-relocation use
16     Prevent multipathing/Suppress devices from VxVM's view
17     Allow multipathing/Unsuppress devices from VxVM's view
18     List currently suppressed/non-multipathed devices
list   List disk information

?      Display help about menu
??     Display help about the menuing system
q      Exit from menus

```

Select an operation to perform:

---

Select option 1 “Add or initialize one or more disks”; a screen appears describing all the possible options of adding the disk. The options are as follows:

- ▶ Add the disk to an already existing disk group
- ▶ Add the disk to a new disk group, therefore creating a new disk group
- ▶ Add the disk without adding it to a disk group

The screen also describes the possibilities of adding the disk for hot relocation. For more information, see 5.5.1, “Hot relocation” on page 167.

At the following prompt:

Select disk devices to add: [<pattern-list>,all,list,q,?]

type in `list` to get a list of disks that are on the system, then at the next prompt type in the disk name to add. A confirmation message will appear; press Enter, and yet another message will appear about the options available on how to add the disk, as described earlier followed by the following prompt:

Which disk group [<group>,none,list,q,?] (default: rootdg)

From here you can list the disk groups on the system by typing `list` and then pressing Enter. The prompt as above will reappear after the disk groups are listed. Type in `none` and then press Enter to add the disk to VxVM without adding it to a disk group. The screen appears as shown in Example 4-5 on page 92.

*Example 4-5 vxdiskadm adding disk confirmation*

---

The disk will be initialized and left free for use as a replacement disk.

hdisk4

Continue with operation? [y,n,q,?] (default: y)

---

Press Enter, and then press Enter again at the following prompt:

Use a default private region length for the disk?  
[y,n,q,?] (default: y)

Press Enter in response to the following prompt:

Add or initialize other disks? [y,n,q,?] (default: n)

**vxdiskadm** will then return to the main menu. Press q and then Enter to exit.

**Adding disks using SMIT**

To add a disk to VxVM using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Disk Administrator -> Add or Initialize disks**, or use the **smitty vxvmdiskadmadd** fast path. The screen shown in Example 4-6 appears.

*Example 4-6 SMIT add disk to VxVM*

---

Add or Initialize disks

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

|                          | [Entry Fields]       |           |
|--------------------------|----------------------|-----------|
| * Disk Device            | <input type="text"/> | +         |
| Force                    | [no]                 | +         |
| * Add to the Disk group? | [no]                 | +         |
| Disk group name          | <input type="text"/> | +         |
| F1=Help                  | F2=Refresh           | F3=Cancel |
| F5=Reset                 | F6=Command           | F7=Edit   |
| F9=Shell                 | F10=Exit             | Enter=Do  |
|                          | F4=List              | F8=Image  |

---

Press F4, and a list of disks appears that can be selected. Use the up or down arrow keys to move the highlighted bar over the disk to be added. Once the disk is highlighted, press Enter. Press Enter again to initialize the disk; this will not

add the disk to any disk group, but leave it available for allocation to a disk group later.

### 4.3.2 Creating disk groups

After the installation of VxVM, a primary disk group named rootdg is created, which is the default disk group for most commands. Other disk groups can be created as needed to group disks by specific purposes. In order to create a disk group, at least one free disk is required to be added to the new disk group.

#### Creating disk groups using the CLI

The **vx dg** command is used for creating new disk groups. To create a new disk group called newdg using hdisk4, run the following command:

```
# vx dg init newdg disk02=hdisk4
```

In the above example, “disk02” is the name that VxVM refers to the disk. This is known as the disk media name.

#### Creating disk groups using VEA

The following steps explain how to create a disk group using the VEA:

- ▶ After logging into the VEA, double-click on the **Disk Groups** icon in the window on the right hand side. A list of disk groups currently on the system will appear.
- ▶ From the menu bar at the top of the screen, select **Actions -> New Dynamic Disk Group**, or, from the right-hand side window, select the **Disk Group** icon and then right-click. A drop-down menu will appear; select New Dynamic Disk Group from the drop-down menu. The new dynamic disk group wizard will appear; click on the **Next** tab at the bottom of the screen. The screen appears as shown in Figure 4-3 on page 94. In this figure, the fields are filled out for a disk group called newdg, which contains hdisk3, which will be known to VxVM as disk03.

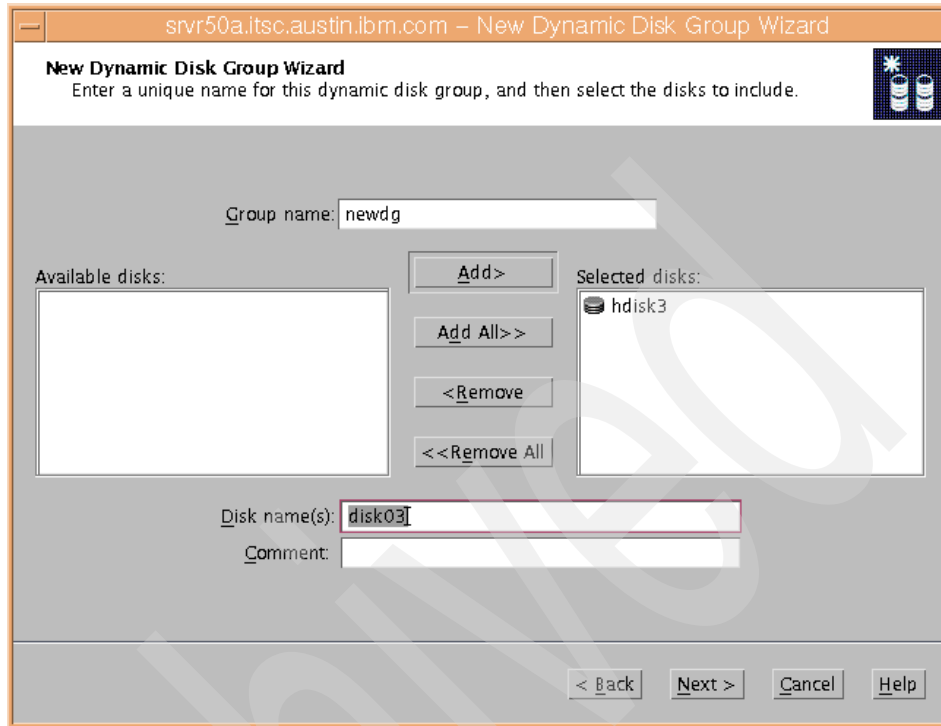


Figure 4-3 VEA add disk group screen

- ▶ In the window that appears (see Figure 4-3), the available disks are listed in a subwindow on the left hand side. At the top will be a field called **Group name**; click on this field and type in the name for the new disk group. Highlight the disk to add to the disk group from the **Available disks** subwindow and click on the **Add** tab. The disk will move from the left-hand **Available disks** window to the right-hand **Selected disks** window. Continue selecting disks that you want to add to the disk group. Disks can be unselected at any time by selecting a disk in the **Selected disks** window and clicking on the **Remove** tab.
- ▶ Once all disks for the disk group have been selected, the disk names can be specified by clicking on the field at the bottom of the screen called **Disk name(s)**. Separate each name by a space. These names will correspond to the disks listed in the **Selected Disks** window from the top to the bottom. If a name is not specified, VxVM will use a default name.
- ▶ Once all details are entered, click on **Next** at the bottom of the screen; a confirmation window will then appear with the selected disks listed. Click on **Yes**; another window will appear showing the specifications of the disk group. Click on **Finish** to create the disk group.

## Creating disk groups using vxdiskadm

Using **vxdiskadm** to create a disk group is much the same process as using **vxdiskadm** to initialize disks in VxVM. To create a disk group, run **vxdiskadm** from the command line; the screen shown in Example 4-4 on page 90 appears. Select option 1, “Add or initialize one or more disks”. At the bottom of the screen, the following prompt will appear:

```
Select disk devices to add: [<pattern-list>,all,list,q,?]
```

Type **list** and then press Enter; a list of disks will appear. Select a free disk from the list and then type in the disk name at the prompt. A confirmation message will appear; press Enter. At the following prompt, type in the new disk group name:

```
Which disk group [<group>,none,list,q,?] (default: rootdg)
```

A message will appear saying there is no disk group of the name that was typed in at the prompt; you will be given the option of creating it. Press Enter at the prompt; the following message appears:

```
Use a default disk name for the disk? [y,n,q,?] (default: y)
```

If **n** is selected here, then a prompt will appear later in the process to rename the disk to be added to the disk group. To rename the disk, type **n** and then press Enter; otherwise, just press Enter. Press Enter at the following prompt:

```
Add disk as a spare disk for test1dg? [y,n,q,?] (default: n)
```

Press Enter again at the following prompt:

```
Exclude disk from hot-relocation use? [y,n,q,?] (default: n)
```

The screen shown in Example 4-7 appears, if **n** was selected for not accepting the default disks name.

### *Example 4-7 vxdiskadm rename disks screen*

---

A new disk group will be created named **newdg** and the selected disks will be added to the disk group with disk names that you will specify interactively.

```
hdisk4
```

```
Continue with operation? [y,n,q,?] (default: y)
```

---

Press Enter, and then press Enter again at the following message:

```
Use a default private region length for the disk?  
[y,n,q,?] (default: y)
```

If n was selected earlier for accepting the default disk names, the following message appears:

```
Enter disk name for hdisk4 [<name>,q,?] (default: newdg01)
```

Type in the new disk name and then press Enter. This will initialize the disk and add it to the new disk group. Press Enter at the next prompt, and **vxdiskadm** will return to the main menu. Press q and then Enter to exit **vxdiskadm**.

**vxdiskadd** can also be used for creating disk groups and uses the same process as the process that adds disks using **vxdiskadm**. Example 4-8 shows the screen output and answers to prompts as a result of running the **vxdiskadd hdisk4** command in order to create a disk group. Note that **vxdiskadd** will echo a warning that the disk group does not exist and will give the option of creating the disk group.

---

*Example 4-8 Using vxdiskadd to create disk groups*

---

```
# vxdiskadd hdisk4
.....
Add or initialize disks
Menu: VolumeManager/Disk/AddDisks

    Here is the disk selected.  Output format: [Device_Name]

    hdisk4

Continue operation? [y,n,q,?] (default: y)

    You can choose to add this disk to an existing disk group, a
    new disk group, or leave the disk available for use by future
    add or replacement operations.  To create a new disk group,
    select a disk group name that does not yet exist.  To leave
    the disk available for future use, specify a disk group name
    of "none".

Which disk group [<group>,none,list,q,?] (default: rootdg) newdg

    There is no active disk group named newdg.

Create a new group named newdg? [y,n,q,?] (default: y)

Use a default disk name for the disk? [y,n,q,?] (default: y)

Add disk as a spare disk for newdg? [y,n,q,?] (default: n)

Exclude disk from hot-relocation use? [y,n,q,?] (default: n)

    A new disk group will be created named newdg and the selected disks
```



will be added to the disk group with default disk names.

hdisk4

Continue with operation? [y,n,q,?] (default: y)

Initializing device hdisk4.

Use a default private region length for the disk?  
[y,n,q,?] (default: y)

Creating a new disk group named newdg containing the disk  
device hdisk4 with the name newdg01.

Goodbye.

---

## Creating disk groups using SMIT

To use SMIT to add a disk group, select the following options: **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> Disk Groups -> Add a Disk Group**, or use the `smitty vxvmdgadd` fast path. The screen shown in Example 4-9 appears. This example shows the required fields already filled out.

*Example 4-9 SMIT add disk group screen*

---

Add a Disk Group

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

\* Disk group name

\* Disk access names

Number of Configuration copies

Number of Log copies

[newdg]

[hdisk2 hdisk3]

[]

[]

+

#

#

F1=Help

F2=Refresh

F3=Cancel

F4=List

F5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do

---

Type in the name of the new disk group in the **Disk group name** field. Use the down arrow key to move to the **Disk access names** field and then press F4; a list of available disks will be displayed. Use the up or down arrow keys to move the highlighted bar over the disk to select. With the disk highlighted, press F7 to

select the disk; this will place a “>” next to the disk; to unselect a disk, press F7 again. Multiple disks can be selected. Once all disks are selected, press Enter, then press Enter again to create the disk group.

### 4.3.3 Creating volumes

Once all disk groups are set up, volumes can be created on the disks within the disk group. Volumes can span multiple disks and many different types of volumes can be created depending on the requirements of the applications that use the file systems mounted on these volumes. The following list shows the type of volumes that can be created under VxVM:

- ▶ Simple concatenated volumes
- ▶ Striped (RAID-0) volumes
- ▶ Mirrored (RAID-1) volumes
- ▶ Striped and mirrored (RAID 0+1) volumes
- ▶ RAID 5 volumes
- ▶ Mirrored and striped volumes (RAID 1+0) volumes

From the command line, there are two commands that can be used to create volumes: they are **vxassist** and **vxmake**. **vxmake** requires a detailed understanding of how volumes are constructed and will be covered in 5.2, “Volume administration” on page 136. **vxassist** does not require the same understanding of how volumes are created and will be covered in this chapter. By default, **vxassist** will create volumes in the rootdg disk group unless the -g option is specified. The syntax of **vxassist** is as follows:

```
# vxassist [-g diskgroup] action volume attributes
```

**vxassist** can also be set up to read from a defaults file. If the file /etc/default/vxassist exists, the values specified in this file will be used by the **vxassist** command. This is useful to avoid typing long **vxassist** commands at the command line. Below is an example of the /etc/default/vxassist file that will create all volumes in the vxdg01 disk group of type striped:

```
# cat /etc/default/vxassist
diskgroup=vxdg01
layout=stripe
```

This will change the default disk group from rootdg to vxdg01 and the default volume layout from concatenated to striped. Refer to the **vxassist** man page for other options that can go in this file.

**Note:** The /etc/default/vxassist file does not exist by default and needs to be created. On AIX, the /etc/default directory may not exist and may need to be created. A different default file can be used by using the -d option with **vxassist**.

All types of volumes can be created via the VEA and from SMIT apart from RAID 1+0 volumes, which cannot be created from SMIT menus. There is not an available VERITAS supplied menu utility to create volumes.

**Important:** If using the VEA to create volumes, make sure the /etc/default/vxassist file does not exist. If it does, rename it, as the VEA will read this file and override the options you have taken in VEA.

### Creating simple concatenated volumes

Simple concatenated volumes are mainly used when a file system is required that is larger than one physical disk, so the underlying volume spans two or more physical disks. Concatenated volumes do not provide any redundancy, so if one disk fails, then data will be lost on the whole volume. Example 4-10 shows an example of the layout of a concatenated volume; it consists of one plex and, in this case, one subdisk, but other examples may contain many subdisks.

*Example 4-10 Concatenated volume example*

```
# vxprint concatvol
Disk group: rootdg
```

| TY | NAME         | ASSOC        | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|--------------|---------|--------|--------|--------|--------|--------|
| v  | concatvol    | fsgen        | ENABLED | 204800 | -      | ACTIVE | -      | -      |
| p1 | concatvol-01 | concatvol    | ENABLED | 204800 | -      | ACTIVE | -      | -      |
| sd | disk02-02    | concatvol-01 | ENABLED | 204800 | 0      | -      | -      | -      |

### Using vxassist to create simple concatenated volumes

To create a simple concatenated volume called newvol of 50 MB in size using **vxassist**, run the following command:

```
# vxassist make newvol 50m
```

The default size for **vxassist** is in 512 KB blocks. The size value can be specified in m for megabytes, k for kilobytes, g for gigabytes, t for terabytes, or the default value of 512 byte blocks.

To create a simple volume in a diskgroup called newdg, use the following command:

```
# vxassist -g newdg make newvol 50m
```

### Using the VEA to create simple concatenated volumes

The following steps are used to create a simple concatenated volume using the VEA:

1. After logging into the VEA, double-click on the **Volumes** icon in the suborned on the right hand side.
2. From the options on the menu bar at the top of the screen, select **Actions** -> **New Volume**; the new volume wizard appears. Click on the next tab at the bottom of the screen. The screen shown in Figure 4-4 appears. In this figure, the necessary fields are filled out in order to create a 50 MB concatenated volume in the rootdg disk group called concatvol.

svr50a.itsc.austin.ibm.com - New Volume Wizard

**New Volume Wizard**  
Select the attributes for this volume.

Group name: rootdg  
Volume name: concatvol  
Comment:  
Size: 50 MB Max Size  
Layout:  
☒ Concatenated  
☐ Striped Columns: 2  
☐ RAID-5 Stripe unit size: 128  
☐ Concatenated Pro  
☐ Striped Pro  
Mirror Info:  
☐ Mirrored  
Total mirrors: 2  
☐ Enable logging  
☐ Initialize zero  
☐ No layered volumes  
Concatenated: A simple volume with a single copy of data on one or more disks.  
< Back Next > Cancel Help

Figure 4-4 VEA create volume screen

3. In the screen shown in Figure 4-4, select the disk group to create the volume in by clicking on the drop-down next to the **Group Name** field. Type in the volume name in the **Volume name** field. An optional comment can go in the **Comment** field. Type in the volume size in the **Size** field. Next to the size field is a drop-down box where the size measurement can be selected; select the appropriate measurement (MB for megabytes, KB for kilobytes, GB for gigabytes, TB for terabytes or Sectors for 512 byte sectors).

4. In the layout suborned, click the check box next to **Concatenated**. Click on **Next**; a window will appear that gives the option of selecting the disks to use for the volume or letting the volume manager decide which disks to use. Click on the check box next to **Let Volume Manager decide what disks to use for this volume** then click on **Next**.
5. A window appears giving the option of creating a file system on this volume. Click on the check box next to **No file system** and then click on **Next**.
6. A confirmation screen appears listing the options taken; click on **Finish** to create the volume, the screen will go back to the volumes window and the new volume will be listed with the other volumes found on the system after a few seconds.

### Using SMIT to create simple concatenated volumes

To use SMIT to create a concatenated volume, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Add a VxVM Volume -> Add a Concatenated VxVM Volume**, or use the `smitty vxvmlvaddconcat` fast path. Example 4-11 shows an example of the screen.

Example 4-11 SMIT add concatenated volume

Add a Concatenated VxVM Volume

Type or select a value for the entry field.  
Press Enter AFTER making all desired changes.

\* Disk Group

[Entry Fields]

F1=Help

F2=Refresh

F3=Cancel

F4=List

F5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do

Press F4 to bring up a list of disk groups that are available on the system. Select the disk group to add the volume to by using the up or down arrow; once the disk group is selected, press Enter. The screen is shown in Example 4-12. In this example, the fields are already filled out for a concatenated volume of 50 MB in size called concatvol.

Example 4-12 SMIT add concatenated volume screen

Add a Concatenated VxVM Volume

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

|   | [Entry Fields] |           |
|---|----------------|-----------|
| * Disk Group                              | rootdg         |           |
| * Name of the Volume                      | [concatvol]    |           |
| * Size of the Volume (in 512 byte blocks) | [100000]       | #         |
| Mirrored                                  | no             | +         |
| Number of Mirrors                         | []             | #         |
| Enable Logging                            | none           | +         |
| Place Volume on Specific Disks            | []             | +         |
| Initialize Zero                           | no             | +         |
| Comment                                   | []             |           |
| F1=Help                                   | F2=Refresh     | F3=Cancel |
| F5=reset                                  | F6=Command     | F7>Edit   |
| F9=Shell                                  | F10=Exit       | Enter=Do  |
|   | F4=List        | F8=Image  |

Type in the name of the volume in the “Name of the Volume field” and then specify the size in the “Size of the Volume” field. The size can only be specified in 512 byte sectors. Press Enter; the volume is then created.

## Creating striped (RAID-0) volumes

Striped volumes are used for increased performance over concatenated volumes. Like concatenated volumes, they do not provide any redundancy. Striped volumes require at least two disks to be effective. The subdisks in a striped volume are often referred to as columns. Example 4-13 shows an example of the layout of a striped volume; it contains one plex consisting of two subdisks and the stripe size is 64 KB. Note the stripe size is in 512 byte blocks.

*Example 4-13 Striped volume example layout.*

```
# vxprint -th stripevol
Disk group: rootdg
```

| V NAME          | RVG          | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX      |
|-----------------|--------------|---------|----------|--------|-----------|---------------|
| UTYPE           |              |         |          |        |           |               |
| PL NAME         | VOLUME       | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID MODE |
| SD NAME         | PLEX         | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE MODE   |
| SV NAME         | PLEX         | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM MODE    |
| DC NAME         | PARENTVOL    | LOGVOL  |          |        |           |               |
| SP NAME         | SNAPVOL      | DCO     |          |        |           |               |
| v stripevol     | -            | ENABLED | ACTIVE   | 102400 | SELECT    | stripevol-01  |
| f               |              |         |          |        |           |               |
| sgen            |              |         |          |        |           |               |
| p1 stripevol-01 | stripevol    | ENABLED | ACTIVE   | 102400 | STRIPE    | 2/128 RW      |
| sd disk04-04    | stripevol-01 | disk04  | 204800   | 51200  | 0/0       | hdisk0 ENA    |
| sd disk01-04    | stripevol-01 | disk01  | 204800   | 51200  | 1/0       | hdisk2 ENA    |

**Using vxassist to create a striped volume**

To create a striped volume called stripevol of 50 MB in a disk group called diskdg01, use the following command:

```
# vxassist -g diskdg01 make stripevol 50m layout=striped
```

**Using the VEA to create a striped volume**

To create a striped volume using the VEA, follow the first three steps listed in “Using the VEA to create simple concatenated volumes” on page 100, then perform the following steps:

- 1. In the suborned labelled **Layout**, click on the check box next to **Striped**; the **Columns** and **Stripe unit size** fields go from being greyed out to active once the **Striped** check box is selected. Click on **Next**.
- 2. A window appears that allows disks to be selected manually for the volume or to let volume manager allocate the disks. Take the default option of letting volume manager allocate the disks by clicking on **Next**.
- 3. A window appears that allows a file system to be created on the volume. Select the default of **No file system** by clicking on **Next**.
- 4. A summary window appears with the options selected. Click on **Finish**, and the volume will then be created and the screen will list all volumes on the system, including the newly created volume, after a few seconds.

**Creating striped volumes using SMIT**

To create a striped volume using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Add a VxVM Volume -> Add a Striped VxVM Volume**, or use the **smitty vxvmlvaddstriped** fast path. Press F4; a list of disk groups appears. Select the disk group to add the volume to by using the up or down arrow keys. Once the disk group is selected, press Enter. The screen shown in Example 4-14 appears. This example shows the required fields already filled out.

*Example 4-14 SMIT add striped volume*

Add a Striped VxVM Volume

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

|   |                |   |
|---|----------------|---|
|   | [Entry Fields] |   |
| * Disk Group                              | rootdg         |   |
| * Name of the Volume                      | [stripevol]    |   |
| * Size of the Volume (in 512 byte blocks) | [100000]       | # |
| * Number of Columns                       | [2]            | # |
| Stripe Unit Size                          | []             | # |
| Mirrored                                  | no             | + |

|                                |                          |   |
|--------------------------------|--------------------------|---|
| Number of Mirrors              | <input type="checkbox"/> | # |
| Enable Logging                 | none                     | + |
| Place Volume on Specific Disks | <input type="checkbox"/> | + |
| Initialize Zero                | no                       | + |
| Comment                        | <input type="checkbox"/> |   |

F1=Help

F2=Refresh

F3=Cancel

F4=List

F5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do

Type in the name of the volume in the “Name of the Volume” field. Type in the size of the volume in the “Size of the Volume” field; this needs to be entered in 512 byte blocks. Type in the number of columns to use in the “Number of columns” field; this is the number of subdisks the stripe will span. Press Enter, and the volume is then created.

## Creating mirrored (RAID 1) volumes

Mirrored or RAID 1 volumes provide a copy of the data being written to the volume. Mirrored volumes can have up to 32 copies of the data being written to them. A copy of the data is known as a plex. Mirrored volumes have at least two plexes while striped and concatenated volumes only have one. In order to create a mirrored volume, at least two disks are required. Example 4-15 shows an example of the layout of a mirrored volume. Mirrored volumes consist of two or more plexes containing subdisks.

*Example 4-15 Mirrored volume layout example*

```
# vxprint -th mirrvol
Disk group: rootdg
```

| V NAME  | RVG       | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX | UTYPE |
|---------|-----------|---------|----------|--------|-----------|----------|-------|
| PL NAME | VOLUME    | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID | MODE  |
| SD NAME | PLEX      | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE   | MODE  |
| SV NAME | PLEX      | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM    | MODE  |
| DC NAME | PARENTVOL | LOGVOL  |          |        |           |          |       |
| SP NAME | SNAPVOL   | DCO     |          |        |           |          |       |

|               |            |         |        |        |        |        |       |
|---------------|------------|---------|--------|--------|--------|--------|-------|
| v mirrvol     | -          | ENABLED | ACTIVE | 102400 | SELECT | -      | fsgen |
| pl mirrvol-01 | mirrvol    | ENABLED | ACTIVE | 102400 | CONCAT | -      | RW    |
| sd disk04-02  | mirrvol-01 | disk04  | 51200  | 102400 | 0      | hdisk0 | ENA   |
| pl mirrvol-02 | mirrvol    | ENABLED | ACTIVE | 102400 | CONCAT | -      | RW    |
| sd disk01-02  | mirrvol-02 | disk01  | 51200  | 102400 | 0      | hdisk2 | ENA   |



**Creating mirrored volumes using vxassist**

To create a mirrored volume called mirrvol in a disk group called vxdg01 of 50 MB, use the following command:

```
# vxassist -g vxdg01 make mirrvol 50m layout=mirror
```

This will create a mirrored volume in which **vxassist** will create two plexes, each one on a separate disk.

**Using the VEA to create mirrored volumes**

To create a mirrored volume using the VEA, follow the first three steps listed in “Using the VEA to create simple concatenated volumes” on page 100, and then perform the following steps:

- 1. In the **Mirror Info** subwindow on the right-hand side, click on the **Mirrored** check box, and then click on **Next**.
- 2. A window appears that gives the option of letting VxVM allocate the disks or manually allocating the disks; take the default option of letting VxVM allocate the disks by clicking on **Next**.
- 3. A screen that gives the option of creating a file system on the volume appears; select the default option of not creating a file system on the volume by clicking on **Next**.
- 4. A summary screen then appears; click on **Finish**, and the volume will then be created. The screen returns to the list of volumes and, after a few seconds, the new volume is listed with the other volumes on the system in the window.

**Using SMIT to create a mirrored volume**

To create a mirrored volume using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Add a VxVM Volume -> Add a Mirrored VxVM Volume**, or use the **smitty vxvmlvaddmirrored** fast path. Press F4; a list of disk groups appears. Use the up or down arrow to select the disk group to add the volume to; once the disk group is selected, press Enter, and the screen shown in Example 4-16 appears. In this example, the fields are filled out for a mirrored volume of 50 MB in length.

*Example 4-16 SMIT add mirrored volume*

Add a Mirrored VxVM Volume

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

\* Disk Group

\* Name of the Volume

\* Size of the Volume (in 512 byte blocks)

[Entry Fields]

rootdg

[mirrvol]

[100000]

#

|                                |                      |   |
|--------------------------------|----------------------|---|
| Layout                         | Concatenated         | + |
| For Striped:                   |                      |   |
| Number of Columns              | <input type="text"/> | # |
| Stripe Unit Size               | <input type="text"/> | # |
| Total Number of Mirrors        | <input type="text"/> | # |
| Enable Logging                 | none                 | + |
| Place Volume on Specific Disks | <input type="text"/> | + |
| Initialize Zero                | no                   | + |
| Comment                        | <input type="text"/> |   |

|          |            |           |          |
|----------|------------|-----------|----------|
| F1=Help  | F2=Refresh | F3=Cancel | F4=List  |
| E5=Reset | F6=Command | F7=Edit   | F8=Image |
| F9=Shell | F10=Exit   | Enter=Do  |          |

---

Type in the name of the volume in the “Name of the Volume” field, and then type in the size of the volume in the “Size of the Volume” field. The size needs to be specified in 512 byte blocks. Once all the details are typed in, press Enter. The volume is then created.

## Creating striped and mirrored (RAID 0+1) volumes

Striped and mirrored volumes combine the performance of striped volumes with the redundancy capabilities of mirrored volumes. At least four disks are required to make a striped and mirrored volume. Example 4-17 shows an example of the layout of a striped mirror. As with all mirrored volumes, it contains two or more plexes and each plex is striped across two subdisks (also known as columns) with a stripe size of 64 KB. Note the stripe size is shown in 512 KB blocks.

*Example 4-17 Striped and mirrored layout example*

```
# vxprint -th stripemir
Disk group: rootdg
```

| V  | NAME         | RVG          | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX | UTYPE |
|----|--------------|--------------|---------|----------|--------|-----------|----------|-------|
| PL | NAME         | VOLUME       | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID | MODE  |
| SD | NAME         | PLEX         | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE   | MODE  |
| SV | NAME         | PLEX         | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM    | MODE  |
| DC | NAME         | PARENTVOL    | LOGVOL  |          |        |           |          |       |
| SP | NAME         | SNAPVOL      | DCO     |          |        |           |          |       |
| v  | stripemir    | -            | ENABLED | ACTIVE   | 102400 | SELECT    | -        | fsgen |
| pl | stripemir-01 | stripemir    | ENABLED | ACTIVE   | 102400 | STRIPE    | 2/128    | RW    |
| sd | disk03-03    | stripemir-01 | disk03  | 204800   | 51200  | 0/0       | hdisk3   | ENA   |
| sd | disk02-03    | stripemir-01 | disk02  | 207680   | 51200  | 1/0       | hdisk4   | ENA   |
| pl | stripemir-02 | stripemir    | ENABLED | ACTIVE   | 102400 | STRIPE    | 2/128    | RW    |
| sd | disk04-05    | stripemir-02 | disk04  | 256000   | 51200  | 0/0       | hdisk0   | ENA   |

### ***Creating striped and mirrored volumes using vxassist***

To create a striped and mirrored volume called stripemirr01, in disk group vxdg01, of 50 MB, use the following command:

```
# vxassist -g vxdg01 make stripemirr01 50m layout=mirror-stripe
```

### ***Creating striped and mirrored volumes using the VEA***

To create a striped and mirrored volume using the VEA, follow the first three steps in “Using the VEA to create simple concatenated volumes” on page 100, then perform the following steps:

1. In the **Layout** subwindow, click the **Striped** check box. In the **Mirror Info** subwindow, click the **Mirrored** check box, and then click on **Next**.
2. A window appears that gives the option of letting VxVM allocate the disks or manually allocating the disks. Take the default option of letting VxVM allocate the disks by clicking on **Next**.
3. A screen that gives the option of creating a file system on the volume appears. Select the default option of not creating a file system on the volume by clicking on **Next**.
4. A summary screen then appears. Click on **Finish**, and the volume will then be created. The screen returns to the list of volumes and, after a few seconds, the new volume is listed with the other volumes on the system in the window.

### ***Creating striped and mirrored volumes using SMIT***

To create a striped and mirrored volume using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Add a VxVM Volume -> Add a Mirrored VxVM Volume**, or use the `smitty vxvmlvaddmirrored` fast path. Press F4; a list of disk groups appears. Use the up or down arrow key to select the disk group to add the volume to; once the disk group is selected, press Enter. Type in the name of the volume in the “Name of the Volume” field and type in the size of the volume in 512 byte blocks in the “Size of the Volume” Field. Arrow down to the “layout” field and press the Tab key; the value will change from Concatenated to Striped. Example 4-18 shows what the screen should look like when creating a striped and mirrored volume of 50 MB called stmirr01.

#### ***Example 4-18 SMIT create striped and mirrored volume***

---

Add a Mirrored VxVM Volume

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

|   | [Entry Fields]           |   |
|---|--------------------------|---|
| * Disk Group                              | rootdg                   |   |
| * Name of the Volume                      | [strmirr01]              |   |
| * Size of the Volume (in 512 byte blocks) | [100000]                 | # |
| Layout                                    | Striped                  | + |
| For Striped:                              |                          |   |
| Number of Columns                         | <input type="checkbox"/> | # |
| Stripe Unit Size                          | <input type="checkbox"/> | # |
| Total Number of Mirrors                   | <input type="checkbox"/> | # |
| Enable Logging                            | none                     | + |
| Place Volume on Specific Disks            | <input type="checkbox"/> | + |
| Initialize Zero                           | no                       | + |
| Comment                                   | <input type="checkbox"/> |   |

|          |            |           |          |
|----------|------------|-----------|----------|
| F1=Help  | F2=Refresh | F3=Cancel | F4=List  |
| F5=Reset | F6=Command | F7=Edit   | F8=Image |
| F9=Shell | F10=Exit   | Enter=Do  |          |

---

Press Enter; the volume is then created.

## Creating RAID 5 volumes

RAID 5 volumes provide both the performance of striping and redundancy of mirroring. In order to create a RAID 5 volume, at least three disks are needed in the disk group. In order to calculate the parity check that RAID 5 requires, an optional additional disk is required for a RAID 5 log. The RAID 5 log provides quicker recovery from a disk failure. Example 4-19 shows an example of a RAID-5 volume. In this example, there are two plexes: one for the volume data and a smaller plex for the RAID 5 log. The RAID 5 log is created by default.

*Example 4-19 RAID 5 example volume layout*

```
# vxprint -th raid5vol
Disk group: rootdg
```

| V NAME         | RVG         | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX | UTYPE |
|----------------|-------------|---------|----------|--------|-----------|----------|-------|
| PL NAME        | VOLUME      | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID | MODE  |
| SD NAME        | PLEX        | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE   | MODE  |
| SV NAME        | PLEX        | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM    | MODE  |
| DC NAME        | PARENTVOL   | LOGVOL  |          |        |           |          |       |
| SP NAME        | SNAPVOL     | DCO     |          |        |           |          |       |
| v raid5vol     | -           | ENABLED | ACTIVE   | 204800 | RAID      | -        | raid5 |
| pl raid5vol-01 | raid5vol    | ENABLED | ACTIVE   | 204800 | RAID      | 3/32     | RW    |
| sd disk04-01   | raid5vol-01 | disk04  | 0        | 51200  | 0/0       | hdisk0   | ENA   |
| sd disk04-03   | raid5vol-01 | disk04  | 153600   | 51200  | 0/51200   | hdisk0   | ENA   |
| sd disk01-01   | raid5vol-01 | disk01  | 0        | 51200  | 1/0       | hdisk2   | ENA   |

|                |             |         |        |        |         |        |     |
|----------------|-------------|---------|--------|--------|---------|--------|-----|
| sd disk01-03   | raid5vol-01 | disk01  | 153600 | 51200  | 1/51200 | hdisk2 | ENA |
| sd disk03-01   | raid5vol-01 | disk03  | 0      | 102400 | 2/0     | hdisk3 | ENA |
| p1 raid5vol-02 | raid5vol    | ENABLED | LOG    | 2880   | CONCAT  | -      | RW  |
| sd disk02-01   | raid5vol-02 | disk02  | 0      | 2880   | 0       | hdisk4 | ENA |

---

### ***Using vxassist to create a RAID 5 volume***

To create a RAID-5 volume called raid5vol of 50 MB in disk group vxdg01, use the following command:

```
# vxassist -g vxdg01 make raid5vol 50m layout=raid5
```

**vxassist** will create one RAID 5 log by default. To create a mirror of the log, use the **-nlog=2** option with the **vxassist** command.

### ***Using the VEA to create a RAID-5 volume***

To create a RAID 5 volume using the VEA, follow the first three steps listed in “Using the VEA to create simple concatenated volumes” on page 100, and then perform the following steps:

1. In the **Layout** subwindow, click on the **RAID 5** check box; notice the **Columns** field value will change from two to three, as RAID 5 volumes require at least three subdisks. Click on **Next**.
2. A window appears that gives the option of letting VxVM allocate the disks or manually allocating the disks. Take the default option of letting VxVM allocate the disks by clicking on **Next**.
3. A screen that gives the option of creating a file system on the volume appears. Select the default option of not creating a file system on the volume by clicking on **Next**.
4. A summary screen then appears; click on **Finish**, and the volume will then be created. The screen returns to the list of volumes and, after a few seconds, the new volume is listed with the other volumes on the system in the volumes window.

### ***Creating RAID 5 volumes using SMIT***

To create a RAID 5 volume using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Add a VxVM Volume -> Add a RAID-5 VxVM Volume**, or use the **smitty vxvmlvaddraid5** fast path. Press F4; a list of disk groups appears; use the up or down arrow key to select the disk group to add the volume to. Once the disk group is selected, press Enter. Type in the name of the volume in the “Name of the Volume” field and type in the size of the volume in 512 byte blocks in the “Size of the Volume” Field. Arrow down to the “Number of Columns” field and type in the number of subdisks the volume will comprise of; this value has to be at least three due to the method RAID 5 uses to calculate parity. Example 4-20

on page 110 shows how the screen should look when creating a RAID 5 volume called raid5vol01 of 50 MB that spans three subdisks.

*Example 4-20 SMIT create RAID-5 volume*

Add a RAID-5 VxVM Volume

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

\* Disk Group

\* Name of the Volume

\* Size of the Volume (in 512 byte blocks)

\* Number of Columns

Stripe Unit Size

Enable Logging

Place Volume on Specific Disks

Initialize Zero

Comment

[Entry Fields]

rootdg

[raid5vo]

[100000]

[3]

[ ]

no

[ ]

no

[ ]

#

#

#

+

+

+

F1=Help

F2=Refresh

F3=Cancel

F4=List

Esc+5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do

Press Enter, and the RAID-5 volume will then be created.

**Creating mirrored and striped (RAID 1+0) volumes**

Mirrored and striped (RAID 1+0) volumes provide all the advantages of mirroring and striping. This type of volume differs from striped and mirrored (RAID 0+1) volumes in that the striping happens over the top of the mirroring. One notable difference between RAID 1+0 volumes and all other types of volumes is that its plexes consist of sub volumes rather than subdisks. This is what is known as a layered volume. RAID 1+0 volumes can be created by using **vxassist** or by using the VEA. At the time of the writing of this redbook, there is not a SMIT menu available to create a RAID 1+0 volume. Example 4-21 on page 111 shows an example of the RAID 1+0 layout and Example 4-22 on page 111 shows the layout of the sub volumes. Note that the **vxprint** command has to be run three times to find the full layout of the volume, as the initial volume's plexes consist of sub volumes rather than subdisks. For more information on the **vxprint** command, see "Displaying volume information using the CLI" on page 118 or the **vxprint** man page.

#### Example 4-21 RAID-1+0 volume layout example

```
# vxprint -th raid10vol
Disk group: rootdg
```

| V NAME                        | RVG           | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX     | UTYPE |
|-------------------------------|---------------|---------|----------|--------|-----------|--------------|-------|
| PL NAME                       | VOLUME        | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID     | MODE  |
| SD NAME                       | PLEX          | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE       | MODE  |
| SV NAME                       | PLEX          | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM        | MODE  |
| DC NAME                       | PARENTVOL     | LOGVOL  |          |        |           |              |       |
| SP NAME                       | SNAPVOL       | DCO     |          |        |           |              |       |
| v raid10vol -                 |               | ENABLED | ACTIVE   | 102400 | SELECT    | raid10vol-03 | fsgen |
| pl raid10vol-03 raid10vol     |               | ENABLED | ACTIVE   | 102400 | STRIPE    | 2/128        | RW    |
| sv raid10vol-S01 raid10vol-03 | raid10vol-L01 | 1       | 51200    | 0/0    | 2/2       | ENA          |       |
| sv raid10vol-S02 raid10vol-03 | raid10vol-L02 | 1       | 51200    | 1/0    | 2/2       | ENA          |       |

#### Example 4-22 RAID10 sub volume layout example

```
# vxprint -th raid10vol-L01
Disk group: rootdg
```

| V NAME                         | RVG           | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX | UTYPE |
|--------------------------------|---------------|---------|----------|--------|-----------|----------|-------|
| PL NAME                        | VOLUME        | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID | MODE  |
| SD NAME                        | PLEX          | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE   | MODE  |
| SV NAME                        | PLEX          | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM    | MODE  |
| DC NAME                        | PARENTVOL     | LOGVOL  |          |        |           |          |       |
| SP NAME                        | SNAPVOL       | DCO     |          |        |           |          |       |
| v raid10vol-L01 -              |               | ENABLED | ACTIVE   | 51200  | SELECT    | -        | fsgen |
| pl raid10vol-P01 raid10vol-L01 |               | ENABLED | ACTIVE   | 51200  | CONCAT    | -        | RW    |
| sd disk01-07                   | raid10vol-P01 | disk01  | 307200   | 51200  | 0         | hdisk2   | ENA   |
| pl raid10vol-P02 raid10vol-L01 |               | ENABLED | ACTIVE   | 51200  | CONCAT    | -        | RW    |
| sd disk02-06                   | raid10vol-P02 | disk02  | 310080   | 51200  | 0         | hdisk4   | ENA   |

```
# vxprint -th raid10vol-L02
Disk group: rootdg
```

| V NAME                         | RVG           | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX | UTYPE |
|--------------------------------|---------------|---------|----------|--------|-----------|----------|-------|
| PL NAME                        | VOLUME        | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID | MODE  |
| SD NAME                        | PLEX          | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE   | MODE  |
| SV NAME                        | PLEX          | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM    | MODE  |
| DC NAME                        | PARENTVOL     | LOGVOL  |          |        |           |          |       |
| SP NAME                        | SNAPVOL       | DCO     |          |        |           |          |       |
| v raid10vol-L02 -              |               | ENABLED | ACTIVE   | 51200  | SELECT    | -        | fsgen |
| pl raid10vol-P03 raid10vol-L02 |               | ENABLED | ACTIVE   | 51200  | CONCAT    | -        | RW    |
| sd disk03-06                   | raid10vol-P03 | disk03  | 307200   | 51200  | 0         | hdisk3   | ENA   |
| pl raid10vol-P04 raid10vol-L02 |               | ENABLED | ACTIVE   | 51200  | CONCAT    | -        | RW    |

### ***Using vxassist to create a mirrored and striped volume***

To create a mirrored and striped volume called raid10vol of 50 MB in disk group vxdg01, use the following command:

```
# vxassist -g vxdg01 make raid10vol 50m layout=stripe-mirror
```

**Important:** The layout option specified when creating RAID-1+0 volumes looks similar to that of creating striped and mirrored (RAID 0+1) volumes. For RAID-1+0 the option should be “layout=stripe-mirror”, and for RAID 0+1, the option should be “layout=mirror-stripe”. This may cause some confusion, causing the wrong volume type being created.

### ***Using the VEA to create a mirrored and striped volume***

To use the VEA to create a mirrored and striped volume, follow the first three steps listed in “Using the VEA to create simple concatenated volumes” on page 100, and then perform the following steps:

1. In the **Layout** subwindow, click on the **Striped Pro** check box; notice that the **Mirrored** check box in the **Mirror Info** subwindow automatically gets selected. Click on **Next**.
2. A window appears that gives you the option of letting VxVM allocate the disks or manually allocating the disks. Take the default option of letting VxVM allocate the disks by clicking on **Next**.
3. A screen that gives you the option of creating a file system on the volume appears. Select the default option of not creating a file system on the volume by clicking on **Next**.
4. A summary screen then appears. Click on **Finish**, and the volume will then be created. The screen returns to the list of volumes and, after a few seconds, the new volume is listed with the other volumes on the system in the volumes window.

## **4.3.4 Viewing VxVM object information**

This section describes the processes of listing and finding out details about various VxVM objects on the system, such as disks, disk groups, and volumes.

### **Listing disk information**

Listing disk information can be useful for finding out the state of the disk, what disk group a disk belongs to, and how many paths there are to the disk, if the disk is under DMP control.



When listing disk information, they can be in any of the following states:

- ▶ Online: The disk is initialized and available to VxVM.
- ▶ Offline: The disk is not available to VxVM.
- ▶ Error: The disk has not been initialized for use with volume manager.
- ▶ Online invalid: the disk was online, but has had a failure.
- ▶ LVM: The disk belongs to the Logical Volume Manager and is not available for use under VxVM.

### ***Using the CLI to find disk information***

The command used for finding out VxVM disk information is **vxdisk** and has the following syntax:

```
# vxdisk [ -g diskgroup ] action argument
```

The **vxdisk list** command can be used to display all disks that are on the system, as shown in Example 4-23. For each disk, this will list information such as the status, VxVM disk name, and the disk group the disk belongs to.

*Example 4-23 Listing disks on a system using vxdisk*

---

```
# vxdisk list
```

| DEVICE | TYPE   | DISK   | GROUP  | STATUS |
|--------|--------|--------|--------|--------|
| hdisk0 | simple | disk04 | rootdg | online |
| hdisk1 | simple | -      | -      | LVM    |
| hdisk2 | simple | disk01 | rootdg | online |
| hdisk3 | simple | disk03 | rootdg | online |
| hdisk4 | simple | disk02 | rootdg | online |

---

The **vxdisk list <diskname>** command can be used to list information about a specific disk, as shown in Example 4-24. With this command, the AIX device name can be used or the VxVM disk name can be used, for example, **vxdisk list disk02** or **vxdisk list hdisk4** will show the same information.

*Example 4-24 Using vxdisk to display disk information*

---

```
# vxdisk list disk02
Device:      hdisk4
devicetag:   hdisk4
type:        simple
hostid:      srvr50a
disk:        name=disk02 id=1029251678.1215.srvr50a
group:       name=rootdg id=1027624662.1025.srvr50a
info:        privoffset=256
flags:       online ready private autoconfig autoimport imported
pubpaths:    block=/dev/vx/dmp/hdisk4 char=/dev/vx/rdmp/hdisk4
version:     2.2
```

```
iosize:    min=512 (bytes) max=512 (blocks)
public:    slice=0 offset=2432 len=17771728
private:   slice=0 offset=256 len=2048
update:    time=1029273644 seqno=0.9
headers:   0 248
configs:   count=1 len=1486
logs:      count=1 len=225
Defined regions:
config    priv 000017-000247[000231]: copy=01 offset=000000 enabled
config    priv 000249-001503[001255]: copy=01 offset=000231 enabled
log       priv 001504-001728[000225]: copy=01 offset=000000 enabled
Multipathing information:
numpaths:  1
hdisk4     state=enabled
```

---

### ***Using the VEA to find disk information***

Follow these steps:

1. After logging into the VEA, double-click on the **Disks** icon in the right-hand subwindow.
2. A subwindow appears, showing the disks on the system, and for each disk, it will list details such as disk name, disk group that the disk belongs and the status of the disk. To find out information about a specific disk, highlight the disk by clicking on it and then, with the disk selected, right-click on the mouse. A drop-down menu appears; select **Properties**.
3. Once the properties have been viewed, click on **OK** to return to the disk view window.
4. To find out what volumes are on each disk, click on **Disk View** above the disks subwindow; a list of all subdisks and volumes that are on these disks appears.

### ***Using vxdiskadm to list disk information***

From the **vxdiskadm** main menu, select “List disk information” by typing in **list** at the menu prompt. The following prompt appears:

```
Enter disk device or "all" [<address>,all,q,?] (default: all)
```

Press Enter to list all the disks on the system. A list of disks on the system is displayed, followed by the following prompt:

```
Device to list in detail [<address>,none,q,?] (default: none)
```

Type in the VXVM disk name to view the details and then press Enter. The details of the disk will then be displayed on a screen similar to that shown in Example 4-24 on page 113, followed by the following prompt:

```
List another disk device? [y,n,q,?] (default: n)
```

Press Enter, press q, and then Enter to exit `vxdiskadm`.

***Listing disk information using SMIT***

To list all disks on a system, select **System Storage Management (Physical & Logical Storage)** -> **VERITAS Volume Manager** -> **VxVM Disk Administrator** -> **List all disks in the System**, or use the `smitty vxvmdiskadm` fast path to get to the **VxVM Disk Administrator** menu. The list of disks on the system is then displayed, as shown in Example 4-25.

*Example 4-25 SMIT list disks on system output*

| COMMAND STATUS   |        |        |        |        |
|--|--------|--------|--------|--------|
| Command: OK                      stdout: yes                      stderr: no                           |        |        |        |        |
| Before command completion, additional instructions may appear below.                                   |        |        |        |        |
| DEVICE   | TYPE   | DISK   | GROUP  | STATUS |
| hdisk0   | simple | disk04 | rootdg | online |
| hdisk1   | simple | -      | -      | LVM    |
| hdisk2   | simple | disk01 | rootdg | online |
| hdisk3   | simple | disk03 | rootdg | online |
| hdisk4   | simple | disk02 | rootdg | online |
| F1=Help                      F2=Refresh                      F3=Cancel                      F6=Command |        |        |        |        |
| F8=Image                      F9=Shell                      F10=Exit                      /=Find       |        |        |        |        |
| n=Find Next  |        |        |        |        |

**Listing disk group information**

Finding disk group information can be useful in finding out what disk groups are on the system, what volumes are in a disk group, and what disks are in a disk group.

***Using the CLI to display disk group information***

The command used to list information about disk groups is `vx dg`. To list all available disk groups on a system, run the following command:

```
# vx dg list
NAME      STATE      ID
rootdg    enabled    1027624662.1025.srvr50a
vxdg01    enabled    1029349113.1695.srvr50a
```

To list information about a particular disk group, run the `vx dg list` command followed by the disk group name, for example, to find information about the `vxdg01` disk group, run the `vx dg list vxdg01` command. Example 4-26 on page 116 shows the output of the previous command.

#### Example 4-26 Listing disk group information using vxvg

```
# vxvg list vxvg01
Group:      vxvg01
dgid:      1029349113.1695.srvr50a
import-id: 0.1698
flags:
version:   90
detach-policy: global
copies:    nconfig=default nlog=default
config:    seqno=0.1027 permlen=1486 free=1484 templen=2 loglen=225
config disk hdisk4 copy 1 len=1486 state=clean online
log disk hdisk4 copy 1 len=225
```

The following command will show all disks, including those that are in a deported disk group:

```
# vxvg -o alldgs list
```

The **vxvg free** command can be used to list the available space for allocating volumes in a disk group, for example, to list the free space in disk group vxvg01, run the following command:

```
# vxvg -g vxvg01 free
```

| DISK   | DEVICE | TAG    | OFFSET | LENGTH   | FLAGS |
|--------|--------|--------|--------|----------|-------|
| vxvg01 | hdisk4 | hdisk4 | 0      | 17771728 | -     |

This example shows each disk in the disk group and the free space available on each disk in the “LENGTH” column. The free space is shown in 512 byte blocks.

#### Listing disk group information using the VEA

Follow these steps:

1. After logging into the VEA, double-click on the **Disk Groups** icon in the right-hand subwindow.
2. A subwindow appears showing the disk groups on the system. For each disk group, it will list details, such as disk group name, size of the disk group, and percentage of the disk group disk space that is in use.
3. To list the disks in the disk group, click on the **Disks** tab above the disk group view window.
4. To list the volumes in the disk group, click on the **Volumes** tab above the disk group view window.
5. To list the free disk space available in the disk group, click on the **Disk View** tab above the disk group window.

### ***Listing disk group information using SMIT***

To list disk group information using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> Disk Groups**, or use the `smitty vxvmdg` fast path. To list all disk groups, select **List All Disk Groups**; all disk groups on the system will be listed. To list information about a disk group, select **List Contents of a Disk Group**; a screen appears prompting for a disk group name. Press F4, and a list of disk groups on the system appears. Select the disk group by using the up or down arrows. Once the disk group is selected, press Enter. Information is displayed about the disk group, as shown in Example 4-26 on page 116.

### **Listing volume information**

Listing volume information may be required for such purposes as finding out the type of volume and what the state of the volume is. There are two states associated with a volume: the kernel state and the volume state. The VxVM software determines the kernel state and cannot be manipulated by an administrator. The following states are possible kernel states the volume could be in:

- ▶ Detached: Maintenance is being performed and the volume cannot be read or written to.
- ▶ Disabled: The volume is offline and cannot be written to or read from.
- ▶ Enabled: The volume is online and fully functional.

The following are the volume states:

- ▶ Active: The volume is started and in use. The kernel state will be enabled.
- ▶ Clean: The volume is not started but is in a clean state to be started. The kernel state will be disabled.
- ▶ Empty: The volume contents are disabled. The kernel state is disabled. This is usually the state of newly created volumes before they are started.
- ▶ Needsync: The volume is not started and when it is started a resynchronization will be required of the plexes or parity for mirrored and RAID-5 volumes. In this state, the kernel state is disabled.
- ▶ Replay: This state only applies to RAID-5 volumes; the volume is in the process of recovering via the RAID-5 log. The kernel state for this is Enabled.
- ▶ Sync: The volume is started and is in the process of updating plex information for mirrored volumes or parity for RAID-5 volumes. The kernel state for this is enabled.

**Displaying volume information using the CLI**

The command for displaying volume properties is **vxprint**. To display properties of a volume called concatvol, use the following command:

```
# vxprint concatvol
Disk group: rootdg

TY NAME      ASSOC      KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
v  concatvol  fsgen      ENABLED 102400  -       ACTIVE -       -
pl concatvol-01 concatvol  ENABLED 102400  -       ACTIVE -       -
sd disk02-02  concatvol-01 ENABLED 102400  0       -       -       -
```

This shows the volume with its associated plexes and subdisks. The size of the volume can be determined by the value in the LENGTH column, where it intersects with the row that shows the volume information. To find out the type of volume, use the -th options with the **vxprint** command, as shown in Example 4-27. From this example, the volume type can be determined by checking the LAYOUT column in the pl (plex) row.

*Example 4-27 vxprint sample output*

```
# vxprint -th concatvol
Disk group: rootdg

V  NAME      RVG      KSTATE  STATE  LENGTH  READPOL  PREFPLEX  UTYPE
PL NAME      VOLUME   KSTATE  STATE  LENGTH  LAYOUT   NCOL/WID  MODE
SD NAME      PLEX     DISK    DISKOFFS LENGTH  [COL/]OFF DEVICE  MODE
SV NAME      PLEX     VOLNAME NVOLLAYR LENGTH  [COL/]OFF AM/NM   MODE
DC NAME      PARENTVOL LOGVOL
SP NAME      SNAPVOL   DCO

v  concatvol  -        ENABLED ACTIVE  102400  SELECT   -         fsgen
pl concatvol-01 concatvol  ENABLED ACTIVE  102400  CONCAT   -         RW
sd disk02-02  concatvol-01 disk02    2880    102400  0        hdisk4    ENA
```

**Listing volume information using the VEA**

After logging into the VEA, double-click on the **Volumes** icon in the right-hand subwindow; a subwindow appears, showing all the volumes on the system, and for each volume, the type of volume will be shown in the far right-hand side column. The disks the volumes consist of can be listed by clicking on the **Disks** tab above the volumes window.

**Listing volume information using SMIT**

Using the SMIT option is not as powerful as using **vxprint** and the VEA to find out volume information. To list volume information using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> List VxVM volumes in all or specific Disk**

**Groups**, or use the **smitty vxvmlvlist** fast path. A screen appears prompting for a disk group; press Enter to list all volumes on the system. Example 4-28 shows an example of the screen after this option has been taken.

*Example 4-28 SMIT list all volumes output*

COMMAND STATUS

Command: OK

stdout: yes

stderr: no

Before command completion, additional instructions may appear below.

Disk group: rootdg

| V | NAME      | RVG | KSTATE  | STATE  | LENGTH | READPOL | PREFPLEX | UTYPE |
|---|-----------|-----|---------|--------|--------|---------|----------|-------|
| v | concatvol | -   | ENABLED | ACTIVE | 102400 | SELECT  | -        | fsgen |
| v | mirrvol   | -   | ENABLED | ACTIVE | 102400 | SELECT  | -        | fsgen |
| v | raid5vol  | -   | ENABLED | ACTIVE | 102400 | RAID    | -        | raid5 |

F1=Help

F2=Refresh

F3=Cancel

F6=Command

F8=Image

F9=Shell

F10=Exit

/=Find

n=Find Next

To find information about a specific volume, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Show the Characteristics of the Volume**, or use the **smitty vxvmlvshow** fast path. A screen appears prompting for a disk group name; press F4, and a list of disk group names appears. Select the disk group name by using the up or down arrow and pressing Enter when the disk group name is selected. Select the volume name by pressing F4; a list of volumes in the selected disk group appears. Select the volume by using the up or down arrow and when the volume is highlighted, pressing Enter. Press Enter, and the volume details will be listed. Example 4-29 shows the screen after this option has been taken.

*Example 4-29 SMIT list volume details*

| COMMAND STATUS   |         |             |         |            |        |            |                |
|--|---------|-------------|---------|------------|--------|------------|----------------|
| Command: OK  |         | stdout: yes |         | stderr: no |        |            |                |
| Before command completion, additional instructions may appear below. |         |             |         |            |        |            |                |
| V  | NAME    | RVG         | KSTATE  | STATE      | LENGTH | READPOL    | PREFPLEX UTYPE |
| v  | mirrvol | -           | ENABLED | ACTIVE     | 102400 | SELECT     | - fsgen        |
| F1=Help  |         | F2=Refresh  |         | F3=Cancel  |        | F6=Command |                |
| F8=Image   |         | F9=Shell    |         | F10=Exit   |        | /=Find     |                |
| n=Find Next  |         |             |         |            |        |            |                |

### 4.3.5 Creating file systems

When a volume has been created, the next step is to create a file system on the volume. VERITAS file systems can be created on an AIX logical volume as well as a VxVM volume. There are three methods available for creating a file system:

- ▶ Using the CLI
- ▶ Using the VEA
- ▶ Using SMIT

The advantage of using the VEA is that a file system can be created as well as a volume in the one task.

#### Creating a file system using the CLI

The **crfs** command can be used for creating a file system on an already existing volume. The **mkfs** command can also be used for creating file systems; however, it is best to use the **crfs** command, as it will update the necessary entries in the `/etc/filesystems` file, whereas the **mkfs** command will not. To create a VxFS file system on a VxVM volume called `concat` vol in disk group `vxvg01` with a mount point of `/newfs`, run the following command:

```
# crfs -v vxfs -d /dev/vx/rdisk/vxvg01/concatvol -m /newfs
version 4 layout
102400 sectors, 51200 blocks of size 1024, log size 1024 blocks
unlimited inodes, largefiles not supported
51200 data blocks, 50096 free data blocks
2 allocation units of 32768 blocks, 32768 data blocks
last allocation unit has 18432 data blocks
```

The `-absize` option can be used to change the block size. The allowed block sizes are 1024 KB, 2048 KB, 4096 KB and 8192 KB. The default block size is 1024 KB.

**Important:** When using the **crfs** command to create VxFS file systems, ensure that the **crfs** command in the `/sbin/helpers/vxfs` is used, as this has options that the `/usr/sbin/crfs` command does not have.

#### Creating a file system using the VEA

Follow these steps:

1. After logging into the VEA, double-click on the **Volumes** icon in the right-hand subwindow; a subwindow appears, showing all the volumes on the system.
2. Highlight the volume to create a file system on by clicking on it, then, from the menu bar across the top, select **Actions -> File System -> New File System**. A window appears in which the file system options can be set, as shown in Figure 4-5 on page 121.



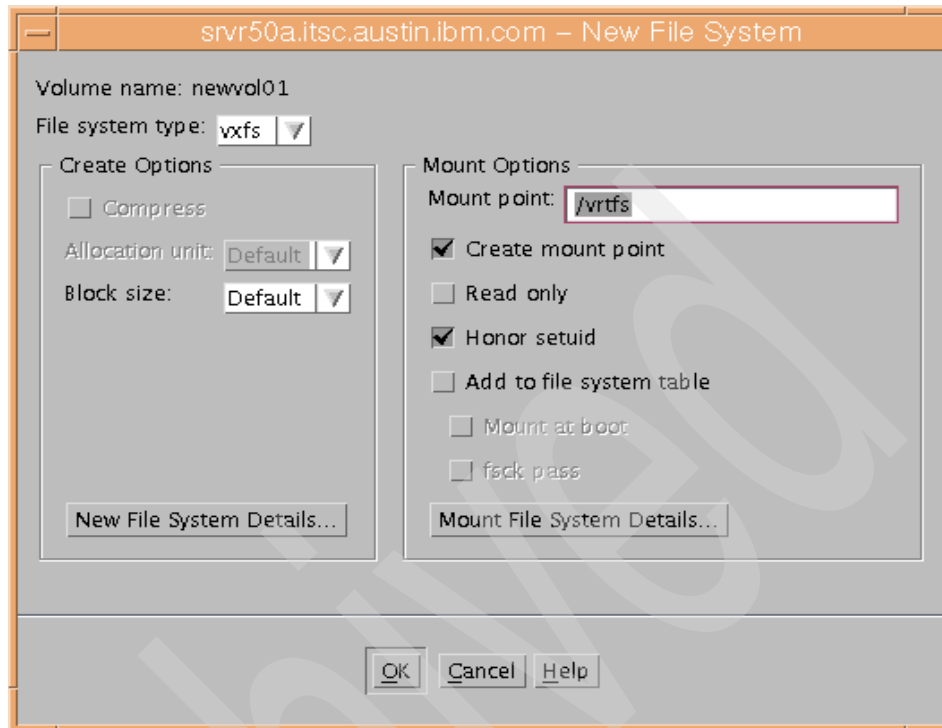


Figure 4-5 VEA create file system window

- ▶ In the **Mount Options** subwindow, type in the mount point in the **Mount Point** field.
- ▶ Click on the **Add to file system table** and **Mount at boot** check boxes, then click on **OK**.
- ▶ The file system is then created. The volume view window reappears and, after a few seconds, the volume details are updated with the file system and mount point columns.

### Creating a file system using SMIT

To create a file system on an already existing VxVM volume under SMIT, select **System Storage Management (Physical & Logical Storage) -> File Systems -> Add/Change/Show/Delete File Systems -> VERITAS File System (VxFS) -> Add a VERITAS File System on a Previously Defined Volume**, or use the **smitty crvxfslvstd** fast path. The “Add a VERITAS file system” screen appears. Press F4, and a list of volumes appears - both VxVM and LVM logical volumes. Select the volume to create the file system on by using the up or down arrow keys and, when it is highlighted, press Enter. Select the disk group by pressing

F4; a list of disk groups appears. Select the disk group that the volume belongs to by using the up or down arrow keys and, when it is highlighted, press Enter. Put in the volume type by pressing the Tab key twice until “Veritas Volume” appears. Type in the mount point in the “MOUNT POINT” field and press Enter. The file system is then created. Example 4-30 shows an example of creating a file system on volume newvol01, which is mounted on /vrtfs.

*Example 4-30 Creating a file system using SMIT*

---

|   |                |           |          |
|---|----------------|-----------|----------|
| Add a VERITAS File System                     |                |           |          |
| Type or select values in entry fields.        |                |           |          |
| Press Enter AFTER making all desired changes. |                |           |          |
| [Entry Fields]                                |                |           |          |
| * VOLUME name                                 | newvol01       |           | +        |
| * Disk/Volume Group                           | rootdg         |           | +        |
| * Volume Type                                 | Veritas Volume |           | +        |
| * MOUNT POINT                                 | [/vrtfs]       |           |          |
| Mount AUTOMATICALLY at system restart?        | no             |           | +        |
| Filesystem Layout Version                     | 4              |           | +        |
| Filesystem Size (default = volume size)       | []             |           |          |
| Block Size                                    | default        |           | +        |
| Inode Size                                    | default        |           | +        |
| Large File Support                            | no             |           | +        |
| Log Size (blocks)                             | []             |           | #        |
| F1=Help                                       | F2=Refresh     | F3=Cancel | F4=List  |
| F5=Reset                                      | F6=Command     | F7=Edit   | F8=Image |
| F9=Shell                                      | F10=Exit       | Enter=Do  |          |

---

## 4.3.6 Mounting file systems

The VERITAS file systems can be mounted with a number of different options as follows:

- ▶ **log:** All changes to the file system are recorded in a file system log before the system returns to the application.
- ▶ **delaylog:** System calls can return to the application before the intent log is updated. This may be used for performance gains at the expense of reliability. This is the default option.
- ▶ **tmplog:** Almost all application system calls are returned before the log is updated. This should only be used when reliability is not an issue. Provides more performance but not much data reliability.
- ▶ **nodatainlog:** This is a hardware dependant option used for disks that do not support bad block revectoring.

- ▶ **blkclear:** Clears the extents before being allocated to another file. This may be used for security purposes in the event of a system failure to ensure the extent does not have data left over from a previous file ending up in another file.
- ▶ **mincache:** This has five sub options and relates to how files are written to disk. **closesync** is the default option.
  - mincache=closesync
  - mincache=direct
  - mincache=dsync
  - mincache=unbuffered
  - mincache=tmpcache
- ▶ **convosync:** This has five sub options and is also related to how files are written to disk.
  - convosync=closesync
  - convosync=delay
  - convosync=direct
  - convosync=dsync
  - convosync=unbuffered
- ▶ **qlog:** Enables quick log, which requires a separate license for this feature.
- ▶ **largefiles** or **nolargefiles:** Enables or disables the large file capability of the file system.

These mount options can be used with the **mount** command by specifying the **-o** option and separating each option with a comma. For example, to mount a file system called **/newfs** with **largefiles** enabled and **log** enabled, run the following command:

```
# mount -V vxfs -o log,nolargefiles /dev/vx/dsk/rootdg/concatvol /newfs
```

These options can be placed in the **/etc/filesystems** file under the **options** field in order to automatically set the options when the file system is mounted. To determine which mount options are in use, use the **mount** command.

## Mounting file systems using the VEA

- ▶ After logging into the VEA, double-click on the **File Systems** icon in the right-hand window. A list of file systems appears; highlight the file system to mount by clicking on the file system.
- ▶ From the menus, select **Actions -> Mount File System**. The default option is to mount the file system with the options listed in the **/etc/filesystems** file; to

mount the file system with options not set in the /etc/filesystems, uncheck the **Mount using options in the file system table** check box by clicking on it, and then click on the **Mount file system details** tab.

- A window appears where the mount options can be typed in. Type in the mount options and click on **OK**; the window disappears. Click on **OK**, and the file system will be mounted.

To see the current mount options, highlight the file system, right-click on the mouse, and then click **Properties** from the drop-down menu. A window appears, which shows the current mount options at the bottom of the screen.

## Mounting file systems using SMIT

To mount a vxfs file system using SMIT, select **System Storage Management (Physical & Logical Storage) -> File Systems -> Add/Change/Show/Delete File Systems -> VERITAS File System (VxFS) -> Mount a VERITAS File System**, or use the `smitty mountvxfs` fast path. The screen shown in Example 4-31 appears. In this example, the fields are filled out for a volume called mirrvol to be mounted on /vrtfs.

*Example 4-31 SMIT mount VxFS file system screen*

---

Mount a VERITAS File System

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

| [TOP]  | [Entry Fields]           |           |
|--|--------------------------|-----------|
| * VOLUME name                                    | /dev/vx/dsk/rootdg/mir>+ |           |
| * DIRECTORY over which to mount                  | [/vrtfs]                 | +         |
| Clear data extents before allocating? (blkclear) | no                       | +         |
| O_SYNC and O_DSYNC caching behavior (convosync)  |                          | +         |
| Set datainlog flag                               |                          | +         |
| Verify Large File Support                        |                          | +         |
| Intent Logging                                   |                          | +         |
| Caching Behavior (mincache)                      |                          | +         |
| Allow access time updates(noatime)               | yes                      | +         |
| Enable/Disable Quick I/O (qio)                   |                          | +         |
| Enable Disk Quotas                               | noquota                  | +         |
| Read/write or read-only (rw/ro)                  | rw                       | +         |
| Snapshot to mount (snapof)                       | []                       |           |
| Snapshot size (snapsize)                         | []                       |           |
| Allow setuid (suid/nosuid)                       | yes                      | +         |
| [BOTTOM]   |                          |           |
| F1=Help  | F2=Refresh               | F3=Cancel |
| F5=Reset   | F6=Command               | F7=Edit   |
| F9=Shell   | F10=Exit                 | Enter=Do  |
|  |                          | F4=List   |
|  |                          | F8=Image  |

---

Press F4, and a list of file systems to mount appears. Select the file system to mount by using the up or down arrow keys and, when the file system is highlighted, press Enter. In the “Directory over which to mount” field, type in the name of the mount point for the file system. The other options in the screen can be set by moving to the appropriate field and pressing the Tab key to move between the options allowed for that particular field. Once all options are selected, press Enter. The file system is then mounted.

### 4.3.7 Resizing file systems

VERITAS file systems on VxVM volumes can be either increased or reduced as needed. In some cases, the underlying volume may need to be increased if it is too small to hold the new file system size.

**Note:** Increasing a file system may fail if the file system is totally full. If this happens, temporarily move some data off the file system, then increase it and move the data back.

#### Increasing a file system using the CLI

There are two options for increasing a file system using the CLI:

- ▶ Increasing the volume first using **vxassist**, then using **fsadm** to increase the file system. In some cases, the volume may not need to be increased, so only the **fsadm** command needs to be run.
- ▶ Using the **vxresize** command to increase the volume and the file system in one step.

The preferred option is to use the **vxresize** command, as this will increase the underlying volume if needed. The maximum size a file system can be increased to can be determined by running the **vxassist** command. For example, to find the maximum size that a file system named /newfs mounted on a volume named concatvol in disk group vxdg01 can be increased, run the following command:

```
# vxassist -g vxdg01 maxgrow concatvol  
Volume concatvol can be extended by 61661184 to 61763584 (30158Mb)
```

The size shown is in 512 byte blocks.

To increase a file system called /newfs mounted on a volume called concatvol that is 50 MB to 60 MB, run the commands shown in Example 4-32 on page 126. The size specified in the **fsadm** command is obtained from the **vxprint** command.

*Example 4-32 Increasing a file system using vxassist and fsadm*

```
# vxassist -g vxdg01 growby concatvol 10m
# vxprint concatvol
Disk group: vxdg01

TY NAME          ASSOC          KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
v  concatvol     fsgen          ENABLED  122880  -        ACTIVE  -        -
p1 concatvol-01  concatvol     ENABLED  122880  -        ACTIVE  -        -
sd disk02-02     concatvol-01  ENABLED  122880  0        -        -        -
# fsadm -b 122880 /newfs
vxfs fsadm: /dev/vx/rdisk/vxdg01/concatvol is currently 102400 sectors - size
will be increased
```

The same file system and volume can be increased or reduced in one step by using the **vxresize** command as follows:

```
# /etc/vx/bin/vxresize -g vxdg01 concatvol 122880
```

## Reducing a file system using the CLI

**Important:** When reducing a file system ensure there is enough space to reduce the file system, or you may lose data on the file system.

To shrink a file system, use either the **vxresize** command, specifying a size smaller than the file system size, or use **fsadm**, to shrink the file system, and then use **vxassist** with the **shrinkby** or **shrinkto** option to reduce the volume. Example 4-33 shows an example of how to use **fsadm** and **vxassist** to reduce a file system called **/newfs** mounted on volume **concatvol** in disk group **vxdg01**.

*Example 4-33 Shrinking a file system using fsadm and vxassist*

```
# vxprint concatvol
Disk group: vxdg01

TY NAME          ASSOC          KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
v  concatvol     fsgen          ENABLED  122880  -        ACTIVE  -        -
p1 concatvol-01  concatvol     ENABLED  122880  -        ACTIVE  -        -
sd disk02-02     concatvol-01  ENABLED  122880  0        -        -        -
# /opt/VRTSvxfs/sbin/fsadm -b 102880 /newfs
vxfs fsadm: /dev/vx/rdisk/vxdg01/concatvol is currently 122880 sectors - size
will be reduced

# vxassist -g vxdg01 shrinkto concatvol 102880
# vxprint concatvol
Disk group: vxdg01

TY NAME          ASSOC          KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
```

|    |              |              |         |        |   |        |   |   |
|----|--------------|--------------|---------|--------|---|--------|---|---|
| v  | concatvol    | fsgen        | ENABLED | 102880 | - | ACTIVE | - | - |
| p1 | concatvol-01 | concatvol    | ENABLED | 102880 | - | ACTIVE | - | - |
| sd | disk02-02    | concatvol-01 | ENABLED | 102880 | 0 | -      | - | - |

## Resizing file systems using the VEA

Follow these steps:

1. After logging into the VEA, double-click on the **File Systems** icon in the right-hand subwindow
2. A subwindow appears, showing all the file systems on the system. Highlight the file system to resize by clicking on it, then, from the menu bar at the top, select **Actions -> Resize File System**.
3. A window, shown in Figure 4-6, appears with the resize options available in three different fields: **Add by**, **Subtract by**, and **New Size**. To increase the file system to its maximum size, click on the **Max Size** button; this places the maximum size the file system can be increased by in the **New Size** field.

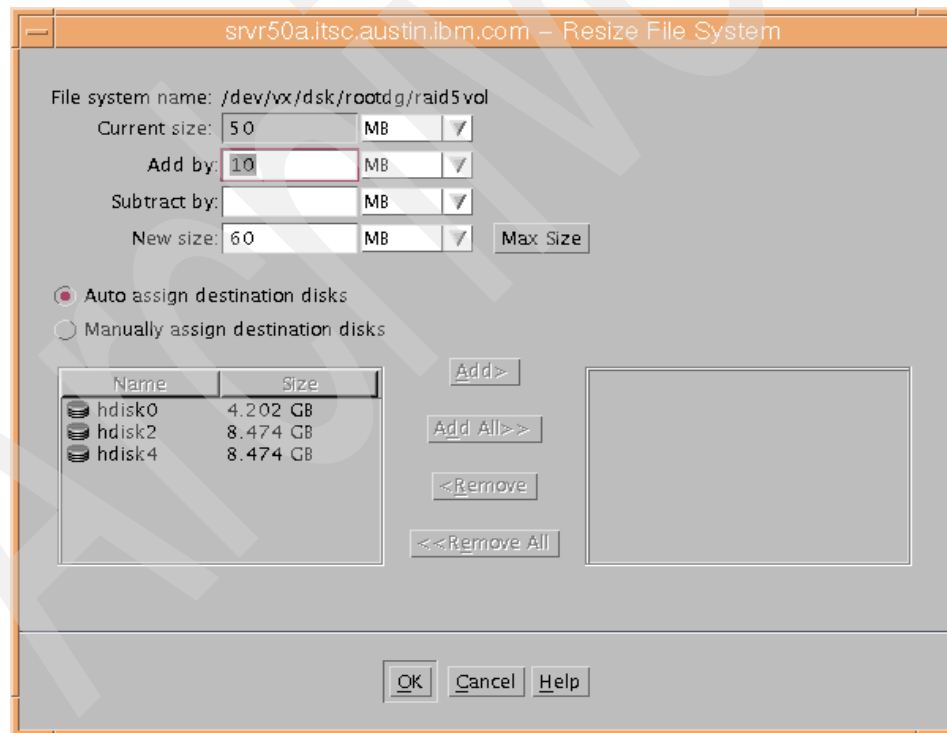


Figure 4-6 VEA resize file system screen

- 4. To increase the file system, input the value to increase the file system by in the **Add by** field, and then select the measurement to increase it by clicking on the drop-down box next to the field (this can be either in 512 byte sectors, KB for kilobytes, MB for megabytes, GB for gigabytes, or TB for terabytes).
- 5. To reduce the file system, place a value to reduce the file system by in the **Subtract by** field. To increase a file system to a certain size, place the new size of the file system in the **New Size** field. If the **Add by** and **Subtract by** fields are updated, then the **New Size** field is automatically updated as well.
- 6. Once the new size of the file system has been entered, click on **OK**. The screen returns to the file system view window and, after a few seconds, the file system details are updated.

Resizing a file system using SMIT

To resize a file system using SMIT, select **System Storage Management (Physical & Logical Storage) -> File Systems -> Add/Change/Show/Delete File Systems -> VERITAS File System (VxFS) -> Change/Show Characteristics of a VERITAS File System**, or use the **smitty vxfs** fast path to get to the **Change/Show Characteristics of a VERITAS File System** menu. A screen is displayed, listing all VxFS file systems; select the file system to resize by using the up or down arrow keys and, when the file system is highlighted, press Enter. The screen shown in Example 4-34 appears.

Example 4-34 SMIT resize file system screen

| Change/Show Characteristics of a VERITAS File System                                    |                |           |          |
|---|----------------|-----------|----------|
| Type or select values in entry fields.<br>Press Enter AFTER making all desired changes. |                |           |          |
| File System Name  | [Entry Fields] |           |          |
| NEW mount point   | /newfs         |           |          |
| SIZE of file system (in sectors)  | [/newfs]       |           |          |
| Mount GROUP   | [123358]       |           |          |
| Mount AUTOMATICALLY at system restart?  | []             |           |          |
| Large File Enabled  | no             |           | +        |
|   | no             |           | +        |
| F1=Help   | F2=Refresh     | F3=Cancel | F4=List  |
| F5=Reset  | F6=Command     | F7=Edit   | F8=Image |
| F9=Shell  | F10=Exit       | Enter=Do  |          |

Using the down arrow key, move to the “SIZE of file system (in sectors)” field and type in the new file system size in 512 byte blocks. Press Enter; the file system is then increased or reduced. The volume is increased as needed, however, if the file system is being reduced when the volume is not reduced as well.



### 4.3.8 Monitoring for failures

The **vxnotify** process is started from the `/etc/init.d/vxvm-recover` startup script at boot time with the `-f` option, which monitors for disk, plex, and volume detach operations. **vxrelocd** or **vxsparecheck**, depending on whether hot relocation or hot sparing is enabled, will check the output from **vxnotify** and send an e-mail to the root user in the event of any problems. To change or add e-mail IDs to be notified, edit the `/etc/init.d/vxvm-recover` script and add or change either one of the following two lines:

```
vxrelocd root &  
vxsparecheck root &
```

Change the root user to another ID or add other IDs after the root user and before the `&` character. Example 4-35 shows the format of the e-mail message received when a failure is detected.

---

*Example 4-35 E-mail format of failure message*

---

Failures have been detected by the VERITAS Volume Manager:

```
failed disks:  
medianame  
failed plexes:  
plexname  
failed log plexes:  
plexname  
failing disks:  
medianame  
failed subdisks:  
subdiskname
```

---

The **vxinfo** command can be used for viewing the current volume and plex states and to determine whether there are any problems. Using the `-p` flag, **vxinfo** will show the state of all plexes and volumes. The `-g <disk group>` flag will show information about volumes and plexes in a disk group; by default, the `rootdg` is shown. For example, to show the state of the plexes and volumes in disk group `vxdbg01`, run the following command:

```
# vxinfo -g vxdbg01 -p  
vol  concatvol      fsgen    Started  
plex concatvol-01   ACTIVE  
vol  mirrvol        fsgen    Started  
plex mirrvol-01     ACTIVE  
plex mirrvol-02     ACTIVE  
vol  newvol         fsgen    Started  
plex newvol-01      ACTIVE
```

The **vxstat** command can be used to report any read and write failures. To list any read or write failures in six second intervals for five seconds on volumes in disk group vxdg01, run the following command:

```
# vxstat -g vxdg01 -ff -i 6 -c 5
FAILED
TYP NAME                READS    WRITES

Thu Aug 15 16:13:11 CDT 2002
vol concatvol           0         0
vol mirrvol             0         0
vol newvol              0         0
vol raid5vol            0         0
.....
```

### Using the VEA to monitor for failures

After logging into the VEA, any detached failures will be displayed in the status area window at the bottom of the screen. For more details, select either one of the following objects - controllers, disk groups, disks, enclosures, or volumes, then click on the alert tab at the top of the right hand side window.

## Advanced administration

In this chapter, we describe more administration tasks that may need to be carried out. This chapter focuses mainly on the command line interface; however, where possible, the SMIT and VEA options are briefly covered. For more information, the following references may be used:

- ▶ *VERITAS Volume Manager 3.2 - Administrator's Guide (AIX)*
- ▶ *VERITAS File System 3.4 - Administrators Guide (AIX)*
- ▶ *VERITAS Volume Manager 3.2 Users Guide - VERITAS Enterprise Administrator User's Guide (AIX)*

In this chapter, the following topics are described:

- ▶ Dynamic Multipathing (DMP)
- ▶ Volume administration
- ▶ Disk group administration
- ▶ Backups
- ▶ Hot sparing and hot relocation
- ▶ Replacing failed disks
- ▶ VERITAS file system features

## 5.1 Dynamic multipathing

Dynamic multipathing (DMP) is enabled by default in VxVM and will be applied to any disk with multiple controllers or other paths to the disk (if the disk is supported under DMP). For information on supported disks and disk arrays, see 3.1.1, “Hardware requirements” on page 42 or the VERITAS Volume Manager hardware notes for AIX. The **vxddladm listsupport** command can be used to list currently supported arrays. For each disk that is multipathed, VxVM combines all the paths into one path called a metanode. All metanodes and disks with single paths can be seen by listing the block device files in the */etc/vx/dmp* directory. Depending on the hardware of the disk, DMP has two multipathing policies for a metanode:

- ▶ **Active/Active:** DMP uses all paths to the disk for I/O balancing; this can provide performance gains. If any of the paths fail, DMP will automatically continue to use the other paths to the disk and re route any I/O from the failed path to another path.
- ▶ **Active/Passive:** DMP only uses one path to the disk for I/O. In the event of a failure, another path will be selected and used for I/O.

To find out what policy is in use, use the **vxdisk** command, as shown in Example 5-1. The multipathing information is displayed at the bottom of the **vxdisk** output. This example shows the output of a disk using the active/active policy. Both paths are enabled; if any of the paths have failed, the state would be disabled.

*Example 5-1 Using vxdisk to show multipathing information*

---

```
# vxdisk list hdisk4
.....
Multipathing information:
numpaths: 2
hdisk4  state=enabled
hdisk6  state=enabled
```

---

Example 5-2 shows an example of using the **vxdisk** command to list information about an active/passive multipathed disk. Note there is an extra field called “type”, which indicates which path to the disk is in use, and which is the backup path. The type field does not appear in active/active disks.

*Example 5-2 Active/Passive multipathing*

---

```
# vxdisk list hdisk8
.....
Multipathing information:
numpaths: 2
hdisk8 state=enabled type=primary
```

---

```
hdisk9 state=enabled type=secondary
```

---

The paths to a disk that are under DMP control can be enabled or disabled by using the **vxdiskadm** utility and selecting either option 16, “Prevent multipathing/Suppress devices from VxVM's view” to remove paths to a disk or option 17, “Allow multipathing/Unsuppress devices from VxVM's view” to enable paths to a disk. Selecting option 16 will give the same menu options as using **vxinstall** to prevent multipathing or suppress devices from VxVM's view. For a description of these options, see “Preventing multipathing or suppressing disks from VxVM” on page 62. Selecting option 17 will give the menu shown in Example 5-3. These options correspond directly with option 16's options and allow the changes performed in option 16 to be undone. Like option 16, “Prevent multipathing/Suppress devices from VxVM's view”, options 1 to 4 are for disk devices that are not supported by DMP and appear as separate disks in the view of VxVM, and options 5 to 7 are for disk devices that support DMP under VxVM. If any changes are made via options 1 to 7, a reboot is required in order for the change to take effect.

*Example 5-3 vxdiskadm allow multipathing/unsuppress devices option*

---

Volume Manager Device Operations  
Menu: VolumeManager/Disk/IncludeDevices

- 1      Unsuppress all paths through a controller from VxVM's view
- 2      Unsuppress a path from VxVM's view
- 3      Unsuppress disks from VxVM's view by specifying a VID:PID combination
- 4      Remove a pathgroup definition
- 5      Allow multipathing of all disks on a controller by VxVM
- 6      Allow multipathing of a disk by VxVM
- 7      Allow multipathing of disks by specifying a VID:PID combination
- 8      List currently suppressed/non-multipathed devices
  
- ?      Display help about menu
- ??    Display help about the menuing system
- q      Exit from menus

Select an operation to perform:

---

## Using vxddmpadm to display and manipulate DMP

The **vxddmpadm** command can be used to find out DMP information and to enable or disable a controller on the system. The following list shows the tasks that can be performed using **vxddmpadm** and an example command used to achieve the task:

- Listing the metanode name that VxVM uses to access a disk:

```
# vxddmpadm getddmpnode nodename=hdisk8
```

| NAME   | STATE   | DA-TYPE | PATHS | ENBL | DSBL | DA-SNO     |
|--------|---------|---------|-------|------|------|------------|
| hdisk6 | ENABLED | JBOD    | 2     | 2    | 0    | JBOD_DISKS |

- ▶ Listing all paths controlled by a DMP metanode; the metanode name is the name returned in the output of the **vxddmpadm getddmpnode** command:

```
# vxddmpadm getsubpaths dmpnodename=hdisk6
```

| NAME   | STATE   | PATH-TYPE | CTLR-NAME | ENCLR-TYPE | ENCLR-NAME |
|--------|---------|-----------|-----------|------------|------------|
| hdisk6 | ENABLED | -         | scsi1     | JBOD       | JBOD_DISKS |
| hdisk8 | ENABLED | -         | scsi2     | JBOD       | JBOD_DISKS |

- ▶ Listing all controllers on the system:

```
# vxddmpadm listctlr all
```

| CTLR-NAME | ENCLR-TYPE  | STATE   | ENCLR-NAME  |
|-----------|-------------|---------|-------------|
| scsi0     | OTHER_DISKS | ENABLED | OTHER_DISKS |
| scsi1     | JBOD        | ENABLED | JBOD_DISKS  |
| scsi2     | JBOD        | ENABLED | JBOD_DISKS  |

- ▶ Listing controllers in an enclosure:

```
# vxddmpadm listctlr enclosure=OTHER_DISKS
```

| CTLR-NAME | ENCLR-TYPE  | STATE   | ENCLR-NAME  |
|-----------|-------------|---------|-------------|
| scsi0     | OTHER_DISKS | ENABLED | OTHER_DISKS |
| scsi2     | OTHER_DISKS | ENABLED | OTHER_DISKS |
| scsi3     | OTHER_DISKS | ENABLED | OTHER_DISKS |
| ssar      | OTHER_DISKS | ENABLED | OTHER_DISKS |

- ▶ Disabling a controller. When a controller is disabled, the command will wait for all I/O to be finished and switched over to another controller on the same disk before returning to the command line. To disable a controller named scsi3, use the following command:

```
# vxddmpadm disable ctlr=scsi3
```

- ▶ Enabling a controller. To enable a controller, run the following command:

```
# vxddmpadm enable ctlr=scsi3
```

## DMP daemons

There are two processes that VxVM uses to monitor and correct DMP problems: the restore daemon and the error daemon. These two processes will not be displayed in the output of the **ps** command as they are kernel processes. The **vxddmpadm** command is used to manipulate these processes. To find out the status of the error daemon, run the following command:

```
# vxddmpadm stat errord
The number of daemons running : 1
```

The error daemon cannot be started or stopped by an administrator and is controlled by VxVM. To check the status of the restore daemon, run the following command:

```
# vxddmpadm stat restored
The number of daemons running : 1
The interval of daemon: 300
The policy of daemon: check_disabled
```

This daemon can be stopped by running the following command:

```
# vxddmpadm stop restore
# vxddmpadm stat restored
The number of daemons running : 0
```

The restore daemon can be started using the following command:

```
# vxddmpadm start restore
```

The restore daemon has two policy modes in which it can run:

- ▶ **check\_disabled**: In this mode, the restore daemon checks for any paths that have been previously disabled due to failure and re-enables them if they are back online. This is the default mode.
- ▶ **check\_all**: In this mode, the restore daemon checks all paths on the system and enables any that are back online.

To change the policy mode, stop the restore daemon, then start it using the policy parameter. To change the policy mode from **check\_disabled** to **check\_all**, run the following commands:

```
# vxddmpadm stop restore
# vxddmpadm start restore policy=check_all
# vxddmpadm stat restored
The number of daemons running : 1
The interval of daemon: 300
The policy of daemon: check_all
```

By default, the restore daemon will check every 300 seconds. To change this interval, use the **interval** parameter with the **vxddmpadm** command. To change the interval, the daemon must be stopped first, then started with the new interval value. For example, to start the restore daemon with an interval of 400 seconds, use the following commands:

```
# vxddmpadm stop restore
# vxddmpadm start restore interval=400
# vxddmpadm stat restored
The number of daemons running : 1
The interval of daemon: 400
The policy of daemon: check_disabled
```

Controllers can be disabled from the VEA by selecting the controller under the controllers view and then selecting **Actions** -> **Disable**.

## 5.2 Volume administration

This section describes how to perform the following volume tasks:

- ▶ Monitoring tasks
- ▶ Create volumes using **vxmake**
- ▶ Adding a mirror to a volume
- ▶ Removing a mirror from a volume
- ▶ Adding and removing a log from a volume
- ▶ Change the layout of an existing volume
- ▶ Creating layered volumes
- ▶ Renaming volumes
- ▶ Removing volumes

### 5.2.1 Monitoring tasks

Long running processes that result from a VxVM command being run can be monitored with the **vxtask** command. This command is useful for monitoring the progress of many volume administration commands. The **vxtask list** command will list every VxVM task running on the system. For each task that is running, a task tag is associated with the task that can be used by the **vxtask** command to monitor a specific task. The following commands have a -t option in which a task ID of up to 16 characters can be used as a parameter, which can be used by the **vxtask** command to monitor.

- ▶ **vxassist**
- ▶ **vxevac**
- ▶ **vxplex**
- ▶ **vxreattach**
- ▶ **vxrecover**
- ▶ **vxresize**
- ▶ **vxsd**
- ▶ **vxvol**

Example 5-4 on page 137 shows an example of monitoring a task. The following steps are performed:

1. A plex is detached from a volume.
2. The **vxrecover** command is run to reattach the plex, specifying a task ID of "recover" with the -t flag.



3. The **vxtask** command is run with the **-i** flag to monitor the task ID specified in the previous step.

*Example 5-4 Monitoring a task*

---

```
# vxplex det mirrvol-02
# vxrecover -b -t recover mirrvol
# vxtask -i recover list
496          PARENT/R  0.00% 1/0(1) VXRECOVER 0.0
497  497      ATCOPY/R 23.01% 0/102400/23560 PLXATT 0.4078 0.4084
```

---

The **vxtask** command can also be used to stop, start, and abort tasks.

Tasks can be monitored from the VEA by clicking on the task tab and right-clicking a task. A pop-up window will appear; select **Properties** from the pop-up window.

## 5.2.2 Creating volumes using vxmake

This section describes how to create volumes using **vxmake**. More control on how a volume is created can be achieved by using **vxmake** as compared to **vxassist**. The following steps are required to create a volume using **vxmake**:

- ▶ Create the subdisks for the volume using **vxmake sd**
- ▶ Combine the subdisks into a plex using **vxmake plex**
- ▶ Associate the plex with a volume using **vxmake vol**
- ▶ Initialize and start the volume using **vxvol start**

The first step in creating a volume using **vxmake** is to create the subdisks (if they do not already exist). Example 5-5 on page 138 shows an example of using **vxmake** with the **sd** parameter to create three subdisks of 50 MB in length. The default size with **vxmake** is in 512 byte sectors; different size values can be specified by using **k** for kilobytes, **m** for megabytes, **g** for gigabytes, and **t** for terabytes. When creating subdisks, **vxmake** will start from the first sector of the disk by default unless the **offset** option is used, as shown by the second **vxmake** command in the example. If an attempt is made to create a subdisk by using sectors that are already allocated to a subdisk, the following warning is issued and the subdisk is unable to be created:

```
vxvm:vxmake: ERROR: creating disk01-04:
      Subdisk disk01-04 would overlap subdisk disk01-01
```

To determine the offset to use when creating a subdisk, use the **vxdg free** command and look in the **offset** and **length** column to determine which offset can be used and how long the length can be.

The third subdisk created in the example shows a shortened form of the command where the disk, offset, and length are used in the command as a comma separated list. Finally, the **vxprint** command with the -s option is run to list the newly created subdisks.

*Example 5-5 Creating subdisks using vxmake*

```
# vxmake sd disk01-01 disk=disk01 len=100000
# vxmake sd disk01-02 disk=disk01 offset=100000 len=100000
# vxmake sd disk01-03 disk01,200000,100000
# vxprint -st
Disk group: rootdg
```

| SD NAME      | PLEX | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE | MODE |
|--------------|------|---------|----------|--------|-----------|--------|------|
| SV NAME      | PLEX | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM  | MODE |
| sd disk01-01 | -    | disk01  | 0        | 100000 | -         | hdisk2 | ENA  |
| sd disk01-02 | -    | disk01  | 100000   | 100000 | -         | hdisk2 | ENA  |
| sd disk01-03 | -    | disk01  | 200000   | 100000 | -         | hdisk2 | ENA  |

Example 5-6 shows an example of how to combine the three subdisks created in Example 5-5 into a plex called newvol01-01 using the **vxmake plex** command. The **vxprint** command with the -p option is then used to display the newly created plex.

*Example 5-6 Creating a plex using vxmake*

```
# vxmake plex newvol01-01 sd=disk01-01,disk01-02,disk01-03
# vxprint -pt
Disk group: rootdg
```

| PL NAME        | VOLUME | KSTATE   | STATE | LENGTH | LAYOUT | NCOL/WID | MODE |
|----------------|--------|----------|-------|--------|--------|----------|------|
| p1 newvol01-01 | -      | DISABLED | -     | 300000 | CONCAT | -        | RW   |

Example 5-7 on page 139 shows how to associate the plex with a volume called newvol01 using the **vxmake** command. When creating a volume using **vxmake**, a usage type parameter must be specified with the -U flag. This usage type parameter determines the default values of a volume by VxVM when created. There are three possible values for Usage type:

- ▶ **fsgen**: Used for volumes with the intention of a file system being created on the volume.
- ▶ **gen**: Used for volumes that do not intend to have file systems created on them.
- ▶ **RAID5**: Used for RAID 5 volumes.

Finally, the volume is initialized and started using the **vxvol start** command. The **vxinfo -p** command is run before and after the volume is initialized to show the state of the volume and its associated plexes.

*Example 5-7 Creating a volume using vxmake and initialization process*

```
# vxmake -Ufsgen vol newvol01 plex=newvol01-01
# vxprint -tvh newvol01
Disk group: rootdg
```

| V NAME  | RVG       | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX | UTYPE |
|---------|-----------|---------|----------|--------|-----------|----------|-------|
| PL NAME | VOLUME    | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID | MODE  |
| SD NAME | PLEX      | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE   | MODE  |
| SV NAME | PLEX      | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM    | MODE  |
| DC NAME | PARENTVOL | LOGVOL  |          |        |           |          |       |
| SP NAME | SNAPVOL   | DCO     |          |        |           |          |       |

```

v newvol01 - DISABLED EMPTY 300000 ROUND -
fsgen
pl newvol01-01 newvol01 DISABLED EMPTY 300000 CONCAT - RW
sd disk01-01 newvol01-01 disk01 0 100000 0 hdisk2 ENA
sd disk01-02 newvol01-01 disk01 100000 100000 100000 hdisk2 ENA
sd disk01-03 newvol01-01 disk01 200000 100000 200000 hdisk2 ENA
# vxinfo -p newvol01
vol newvol01 fsgen Unstartable
plex newvol01-01 EMPTY
# vxvol start newvol01
# vxinfo -p newvol01
vol newvol01 fsgen Started
plex newvol01-01 ACTIVE

```

## Specifying a description file with vxmake

A text file containing the layout of a volume can be specified by using the **-d** option with the **vxmake** command.

Example 5-8 on page 140 shows an example of creating a volume called vol01 in disk group vxdg01 from a description file. The following steps are performed:

- ▶ The description file is displayed showing the layout of a mirrored volume with two plexes with each plex having one subdisk.
- ▶ The **vxmake** command is run specifying the **-d** option.
- ▶ The new volume is started using the **vxvol start** command.
- ▶ The **vxprint** command is run to show the layout of the new volume.

### Example 5-8 Creating a volume from a description file

```
# cat volfile
sd disk02-05 disk=disk02 offset=102400 len=100000
sd disk03-05 disk=disk03 offset=204833 len=100000
plex vol01-01 layout=concat sd=disk02-05
plex vol01-02 layout=concat sd=disk03-05
vol vol01 usetype=fsgen plex=vol01-01,vol01-02
# vxmake -g vxdg01 -d volfile
# vxvol start vol01
# vxprint vol01
Disk group: vxdg01
```

| TY | NAME      | ASSOC    | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|-----------|----------|---------|--------|--------|--------|--------|--------|
| v  | vol01     | fsgen    | ENABLED | 100000 | -      | ACTIVE | -      | -      |
| pl | vol01-01  | vol01    | ENABLED | 100000 | -      | ACTIVE | -      | -      |
| sd | disk02-05 | vol01-01 | ENABLED | 100000 | 0      | -      | -      | -      |
| pl | vol01-02  | vol01    | ENABLED | 100000 | -      | ACTIVE | -      | -      |
| sd | disk03-05 | vol01-02 | ENABLED | 100000 | 0      | -      | -      | -      |

The output from the **vxprint** command can be used in a description file. This is a quick way to reconstruct existing volumes in the event of a total system failure.

## 5.2.3 Adding a mirror to a volume

Existing volumes can be mirrored by attaching another plex to the volume using either the **vxassist** command or by creating a plex using **vxmake** and attaching the plex to the volume via the **vxplex** command. To mirror the volume newvol01 created in Example 5-7 on page 139 using **vxassist**, run the following command:

```
# vxassist mirror newvol01
```

If the volume is in a disk group other than the rootdg, the -g option needs be specified. For example, to mirror a volume called newvol01 in disk group vxdg01, use the following command:

```
# vxassist -g vxdg01 mirror newvol01
```

To use the **vxmake** command and the **vxplex** command to attach a mirror to a volume, first create the subdisk or subdisks on a different disk than which the existing volume is on. Then create the plex using the **vxmake** command.

**Important:** When creating a plex using **vxmake**, ensure it is large enough to hold a mirrored copy of the plex already associated with the volume. The following warning message will be displayed if a plex that is smaller than the plex it is mirroring is attempted to be attached to a volume:

```
vxvm:vxplex: ERROR: Plex newvol01-02 would be a sparse plex of Volume
newvol01
      use -o force to force the operation
```

Example 5-9 shows an example of mirroring a volume by using **vxmake** to create the subdisks and the plex, then attaching the plex to the volume using **vxplex**. The **vxprint** command is then run to show the new volume layout. When the **vxplex** command is run, it may take a while for it to return to the command line, as it needs to synchronize the data from the existing plex onto the newly attached plex. The **vxtask** command can be used to view the progress of the plex attach, as shown in Example 5-10 on page 142. In this example, it can be seen that the procedure is 35% completed.

*Example 5-9 Mirroring a volume using vxmake and vxplex*

```
# vxmake sd disk04-01 disk=disk04 len=300000
# vxplex att newvol01 newvol01-02
# vxprint -tvh newvol01
Disk group: rootdg
```

| V NAME         | RVG         | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX |      |
|----------------|-------------|---------|----------|--------|-----------|----------|------|
| UTYPE          |             |         |          |        |           |          |      |
| PL NAME        | VOLUME      | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID | MODE |
| SD NAME        | PLEX        | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE   | MODE |
| SV NAME        | PLEX        | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM    | MODE |
| DC NAME        | PARENTVOL   | LOGVOL  |          |        |           |          |      |
| SP NAME        | SNAPVOL     | DCO     |          |        |           |          |      |
| v newvol01     | -           | ENABLED | ACTIVE   | 300000 | ROUND     | -        |      |
| fsgen          |             |         |          |        |           |          |      |
| pl newvol01-01 | newvol01    | ENABLED | ACTIVE   | 300000 | CONCAT    | -        | RW   |
| sd disk01-01   | newvol01-01 | disk01  | 0        | 100000 | 0         | hdisk2   | ENA  |
| sd disk01-02   | newvol01-01 | disk01  | 100000   | 100000 | 100000    | hdisk2   | ENA  |
| sd disk01-03   | newvol01-01 | disk01  | 200000   | 100000 | 200000    | hdisk2   | ENA  |
| pl newvol01-02 | newvol01    | ENABLED | ACTIVE   | 300000 | CONCAT    | -        | RW   |
| sd disk04-01   | newvol01-02 | disk04  | 0        | 300000 | 0         | hdisk0   | ENA  |

*Example 5-10 Using vxtask to view plex attach progress*

```
# vxtask list
TASKID PTID TYPE/STATE PCT PROGRESS
213 ATCOPY/R 35.46% 0/300000/106392 PLXATT newvol01 newvol01-02
```

Mirrors can be added to a volume via the VEA by selecting the volume in the volume view and then selecting **Actions -> Mirror -> Add**.

Mirrors can be added to a volume via SMIT by selecting **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Set Characteristics of a VxVM Volume -> Mirror the Volume**, or use the **smitty vxvmlvmirror** fast path.

### 5.2.4 Removing a mirror from a volume

The **vxplex** command with the **dis** option can be used to remove a mirror from a volume. Example 5-11 shows how to remove the mirror that was attached in Example 5-9 on page 141 using **vxplex dis**. The **-g** option is used to specify the disk group the plex is in; the **rootdg** is the default disk group. The **vxprint** command is then run to show the new volume layout.

*Example 5-11 Removing a mirror from a volume*

```
# vxplex -g rootdg dis newvol01-02
# vxprint -tvh newvol01
Disk group: rootdg
```

| V NAME         | RVG         | KSTATE  | STATE    | LENGTH | READPOL   | PREFPLEX      |
|----------------|-------------|---------|----------|--------|-----------|---------------|
| UTYPE          |             |         |          |        |           |               |
| PL NAME        | VOLUME      | KSTATE  | STATE    | LENGTH | LAYOUT    | NCOL/WID MODE |
| SD NAME        | PLEX        | DISK    | DISKOFFS | LENGTH | [COL/]OFF | DEVICE MODE   |
| SV NAME        | PLEX        | VOLNAME | NVOLLAYR | LENGTH | [COL/]OFF | AM/NM MODE    |
| DC NAME        | PARENTVOL   | LOGVOL  |          |        |           |               |
| SP NAME        | SNAPVOL     | DCO     |          |        |           |               |
| v newvol01     | -           | ENABLED | ACTIVE   | 300000 | ROUND     | -             |
| fsgen          |             |         |          |        |           |               |
| pl newvol01-01 | newvol01    | ENABLED | ACTIVE   | 300000 | CONCAT    | - RW          |
| sd disk01-01   | newvol01-01 | disk01  | 0        | 100000 | 0         | hdisk2 ENA    |
| sd disk01-02   | newvol01-01 | disk01  | 100000   | 100000 | 100000    | hdisk2 ENA    |
| sd disk01-03   | newvol01-01 | disk01  | 200000   | 100000 | 200000    | hdisk2 ENA    |

Mirrors can be removed from a volume via the VEA by selecting the volume in the volume view and then selecting **Actions -> Mirror -> Remove**.

Mirrors can be added to a volume via SMIT by selecting **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Remove a Mirror**, or use the `smitty vxvmlremovemirror` fast path.

### 5.2.5 Adding a log to a volume

There are three types of logs that can be added to volumes in order to speed up the recovery process in the event of a volume or plex failure. They are Dirty Region Logging (DRL) for mirrored volumes, RAID 5 logs for RAID 5 volumes, and Data Change Object (DCO) logs. DRLs can be added to non mirrored volumes apart from RAID 5 volumes, but they are most effective in speeding up the resynchronization time of plexes in mirrored volumes. DCO logs are used with FastReync, which requires an additional license. FastReync enables a quicker resync time for plexes and is particularly effective with volume snapshot backups.

#### Adding or removing dirty region logs

To add a dirty region log to an already existing mirrored volume, use the `vxassist` command. Example 5-12 shows an example of adding a log to a mirrored volume. The following steps are performed:

- ▶ The `vxprint` command is run to show the existing layout of the volume.
- ▶ The `vxassist` command is run to add two DRLs to the volume by specifying the `nlog=2` option with the `vxassist` command. This will mirror the DRL.
- ▶ The `vxprint` command is then run to show the layout of the new volume with the two new log plexes.

*Example 5-12 Adding a log to a mirrored volume*

```
# vxprint mirrvol
Disk group: vxdg01
```

| TY | NAME       | ASSOC      | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|------------|------------|---------|--------|--------|--------|--------|--------|
| v  | mirrvol    | fsgen      | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | mirrvol-01 | mirrvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk02-01  | mirrvol-01 | ENABLED | 102400 | 0      | -      | -      | -      |
| pl | mirrvol-02 | mirrvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk04-02  | mirrvol-02 | ENABLED | 102400 | 0      | -      | -      | -      |

```
# vxassist addlog mirrvol logtype=drl nlog=2
# vxprint mirrvol
Disk group: vxdg01
```

| TY | NAME       | ASSOC   | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|------------|---------|---------|--------|--------|--------|--------|--------|
| v  | mirrvol    | fsgen   | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | mirrvol-01 | mirrvol | ENABLED | 102400 | -      | ACTIVE | -      | -      |

|               |            |         |         |     |        |   |   |
|---------------|------------|---------|---------|-----|--------|---|---|
| sd disk02-01  | mirrvol-01 | ENABLED | 102400  | 0   | -      | - | - |
| pl mirrvol-02 | mirrvol    | ENABLED | 102400  | -   | ACTIVE | - | - |
| sd disk04-02  | mirrvol-02 | ENABLED | 102400  | 0   | -      | - | - |
| pl mirrvol-03 | mirrvol    | ENABLED | LOGONLY | -   | ACTIVE | - | - |
| sd disk03-02  | mirrvol-03 | ENABLED | 33      | LOG | -      | - | - |
| pl mirrvol-04 | mirrvol    | ENABLED | LOGONLY | -   | ACTIVE | - | - |
| sd disk04-01  | mirrvol-04 | ENABLED | 33      | LOG | -      | - | - |

To remove a DRL, use the **vxassist** command, for example, to remove the logs added in Example 5-12 on page 143, run the following command:

```
# vxassist remove log mirrvol nlog=2
```

The VEA can be used for adding DRL logs to mirrored volumes only by selecting the volume in the volume view window and selecting **Actions -> Log -> Add**. To remove a log, select **Actions -> Log -> Remove**.

To add a DRL log using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Set Characteristics of a VxVM Volume -> Add log to the volume**, or use the **smitty vxvmlvaddlog** fast path.

To remove a DRL log via smitty, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Remove a Log**, or use the **smitty vxvmlvremove1og** fast path.

## Adding or removing RAID 5 logs

Only one data plex can be associated with RAID 5 volumes. Any attempt to add a plex to a RAID 5 volume will result in the plex being a RAID 5 log. The **vxassist** command is used to attach a RAID 5 log to a RAID 5 volume. By default, one RAID 5 log is added to a RAID 5 volume; however, more RAID 5 logs may be added in order to mirror the current log.

Example 5-13 shows an example of adding a RAID 5 log to a RAID 5 volume. The volume already has a log, as shown in the output of the **vxprint** command, so adding another one will mirror the existing log.

*Example 5-13 Adding a RAID 5 log to a volume*

```
#vxprint raid5vol
Disk group: veritasdg
```

| TY | NAME         | ASSOC       | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|-------------|---------|--------|--------|--------|--------|--------|
| v  | raid5vol     | raid5       | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | raid5vol-01  | raid5vol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | veritas04-01 | raid5vol-01 | ENABLED | 51200  | 0      | -      | -      | -      |
| sd | veritas05-01 | raid5vol-01 | ENABLED | 51200  | 0      | -      | -      | -      |



```
sd veritas06-01 raid5vol-01 ENABLED 51200 0 - - -
pl raid5vol-02 raid5vol ENABLED 2880 - LOG - -
sd veritas01-02 raid5vol-02 ENABLED 2880 0 - - -
#vxassist addlog raid5vol
#vxprint raid5vol
Disk group: veritasdg
```

| TY | NAME         | ASSOC       | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|-------------|---------|--------|--------|--------|--------|--------|
| v  | raid5vol     | raid5       | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | raid5vol-01  | raid5vol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | veritas04-01 | raid5vol-01 | ENABLED | 51200  | 0      | -      | -      | -      |
| sd | veritas05-01 | raid5vol-01 | ENABLED | 51200  | 0      | -      | -      | -      |
| sd | veritas06-01 | raid5vol-01 | ENABLED | 51200  | 0      | -      | -      | -      |
| pl | raid5vol-02  | raid5vol    | ENABLED | 2880   | -      | LOG    | -      | -      |
| sd | veritas01-02 | raid5vol-02 | ENABLED | 2880   | 0      | -      | -      | -      |
| pl | raid5vol-03  | raid5vol    | ENABLED | 2880   | -      | LOG    | -      | -      |
| sd | veritas02-02 | raid5vol-03 | ENABLED | 2880   | 0      | -      | -      | -      |

To remove the log shown in Example 5-13 on page 144, run the following command:

```
# vxassist remove log raid5vol
```

The VEA can be used for adding RAID 5 logs to RAID 5 volumes only. Select the volume in the volume view window and selecting **Actions -> Log -> Add**. To remove a log, select **Actions -> Log -> Remove**. This is the same process as adding a DRL log to a mirrored volume, however, VEA will add a RAID 5 log to a RAID 5 volume.

To add a RAID 5 log using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Set Characteristics of a VxVM Volume -> Add log to the volume**, or use the **smitty vxvmlvaddlog** fast path. This is the same process as adding a DRL log to a mirrored volume however a RAID 5 log is added by SMIT.

To remove a RAID 5 log via SMIT select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Remove a Log** or the fast path **smitty vxvmlvremove log** can be used

## Adding a DCO log to a volume

DCO logs can be added to all volume types apart from RAID 5 volumes.

Example 5-14 on page 146 shows an example of adding a DCO log to a volume. The following steps are performed:

1. The **vxprint** command is run to show the layout of the volume before the log is added.

2. The DCO log is added using the **vxassist** command with a **logtype=dco** parameter.
3. The **vxprint** command is run to show the layout of the volume after the DCO log has been added. By default, the DCO log contains two plexes; however, it is recommended to have one DCO plex per data plex on the volume. The number of DCO plexes can be specified by using the **ndcolog=** option with the **vxassist** command.

*Example 5-14 Adding a DCO log to a volume*

```
# vxprint newvol
Disk group: vxdg01
```

| TY | NAME      | ASSOC     | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|-----------|-----------|---------|--------|--------|--------|--------|--------|
| v  | newvol    | fsgen     | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | newvol-01 | newvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk04-03 | newvol-01 | ENABLED | 102400 | 0      | -      | -      | -      |
| pl | newvol-02 | newvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk03-03 | newvol-02 | ENABLED | 102400 | 0      | -      | -      | -      |

```
# vxassist addlog newvol logtype=dco
# vxprint newvol
Disk group: vxdg01
```

| TY | NAME          | ASSOC         | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|---------------|---------------|---------|--------|--------|--------|--------|--------|
| v  | newvol        | fsgen         | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | newvol-01     | newvol        | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk04-03     | newvol-01     | ENABLED | 102400 | 0      | -      | -      | -      |
| pl | newvol-02     | newvol        | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk03-03     | newvol-02     | ENABLED | 102400 | 0      | -      | -      | -      |
| dc | newvol_dco    | newvol        | -       | -      | -      | -      | -      | -      |
| v  | newvol_dcl    | gen           | ENABLED | 132    | -      | ACTIVE | -      | -      |
| pl | newvol_dcl-01 | newvol_dcl    | ENABLED | 132    | -      | ACTIVE | -      | -      |
| sd | disk02-02     | newvol_dcl-01 | ENABLED | 132    | 0      | -      | -      | -      |
| pl | newvol_dcl-02 | newvol_dcl    | ENABLED | 132    | -      | ACTIVE | -      | -      |
| sd | disk03-02     | newvol_dcl-02 | ENABLED | 132    | 0      | -      | -      | -      |

To remove the DCO log, as shown in Example 5-14, run the following command:

```
# vxdco -g vxdg01 -o rm dis newvol_dco
```

This disassociates the DCO log from the volume and, with the **-o rm** option, it will also be deleted.

To add a DCO log using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Set Characteristics of a VxVM Volume -> Add log to the volume**, or use the **smitty vxvmlvaddlog** fast path. Once in the menu, use the tab key to change the “Log Type” field to DCM.

To remove a DCO log via SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Remove a Log**, or use the `smitty vxvmlvremove log` fast path.

## 5.2.6 Creating layered volumes

This section describes how to create a layered volume from the “ground up” by creating the subdisks, plexes, subvolumes, and finally combining all these into a volume. The key concept of a layered volume is that it uses other volumes called sub volumes rather than subdisks to construct its plexes. This section will use an example of how to create a RAID 1+0 volume called `raid10vol` of 50 MB. This is a lengthy process, and the same result can be achieved much easier by using either `vxassist`, the VEA, or SMIT to create the volume; however, more control on the allocation of disk space can be achieved using this process.

The first part of the process is to create the subdisks on four different disks. Example 5-15 shows an example of how to use `vxmake` to create these four subdisks. Each subdisk is 50 MB in length.

*Example 5-15 Using vxmake to create the subdisks for a RAID 1+0 volume*

---

```
# vxmake sd disk01-04 disk=disk01 offset=400000 len=100000
# vxmake sd disk02-04 disk=disk02 offset=400000 len=100000
# vxmake sd disk03-04 disk=disk03 offset=400000 len=100000
# vxmake sd disk04-04 disk=disk04 offset=400000 len=100000
```

---

After the subdisks have been created, each subdisk needs to be placed into a plex. Example 5-16 shows the commands used to create the plexes.

*Example 5-16 Placing each subdisk into a plex*

---

```
# vxmake plex raid10vol-p01 sd=disk01-04
# vxmake plex raid10vol-p02 sd=disk02-04
# vxmake plex raid10vol-p03 sd=disk03-04
# vxmake plex raid10vol-p04 sd=disk04-04
```

---

After each subdisk is placed into a plex, the subvolumes are created by selecting two of the plexes created in Example 5-16 and creating two volumes using these plexes. The volumes are then started. Example 5-17 shows the commands used to create and start the subvolumes.

*Example 5-17 Creating and starting the subvolumes of a RAID 1+0 volume*

---

```
# vxmake -Ufsgen vol raid10vol-l01 plex=raid10vol-p01
# vxmake -Ufsgen vol raid10vol-l02 plex=raid10vol-p02
# vxvol start raid10vol-l01
```

---

```
# vxvol start raid10vol-102
```

---

After the subvolumes are created, the two remaining plexes are then attached to the volumes in order to mirror them; one plex is attached to each volume, as shown in Example 5-18.

*Example 5-18 Attaching plexes to the RAID 1+0 subvolumes*

---

```
# vxplex att raid10vol-101 raid10vol-p03
# vxplex att raid10vol-102 raid10vol-p04
```

---

After the plexes are attached to the subvolumes, the subvolumes need to have the layered attribute set for VxVM to recognize the volumes as being subvolumes by using the **vxedit set** command, as shown in Example 5-19.

*Example 5-19 Setting the subvolume attribute*

---

```
# vxedit set layered=on raid10vol-101
# vxedit set layered=on raid10vol-102
```

---

The subvolumes created in Example 5-19 are now used to create subdisks for the RAID 1+0 volume, and then placed in a striped plex with two columns, as shown in Example 5-20.

*Example 5-20 Creating subdisks from a subvolume*

---

```
# vxmake sd raid10vol-s01 disk=raid10vol-101 len=100000
# vxmake sd raid10vol-s02 disk=raid10vol-102 len=100000
# vxmake plex raid10vol-01 sd=raid10vol-s01,raid10vol-s02 layout=stripe \
    stwidth=128k ncolumn=2
```

---

Finally, the volume is then created and started, as shown in Example 5-21. The **vxprint** command is then run to show the layout of the RAID 1+0 volume.

*Example 5-21 Creating and starting the RAID 1+0 volume*

---

```
# vxmake -Ufsgen vol raid10vol plex=raid10vol-01
# vxvol start raid10vol
# vxprint raid10vol
Disk group: rootdg
```

| TY | NAME          | ASSOC        | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|---------------|--------------|---------|--------|--------|--------|--------|--------|
| v  | raid10vol     | fsgen        | ENABLED | 199840 | -      | ACTIVE | -      | -      |
| p1 | raid10vol-01  | raid10vol    | ENABLED | 200096 | -      | ACTIVE | -      | -      |
| sv | raid10vol-s01 | raid10vol-01 | ENABLED | 100000 | 0      | -      | -      | -      |
| sv | raid10vol-s02 | raid10vol-01 | ENABLED | 100000 | 0      | -      | -      | -      |

---

## 5.2.7 Changing volume layouts

The layout of an existing volume can be changed while the volume is online and with file systems mounted on the volume. The **vxassist** command is used to change the layout of a volume. There are two types of conversions possible when changing the layout of a volume:

- ▶ Changing the layout of a non-layered volume to another non layered volume layout: Use **vxassist relayout** for this type of conversion.
- ▶ Changing the layout of a volume from layered to non layered or from non layered to layered: Use **vxassist convert** for this type of conversion.

Example 5-22 shows a relayout of a simple concatenated volume to a striped volume with two columns.

*Example 5-22 Changing the layout from concatenated to striped*

---

```
# vxprint -tvh concatvol
.....
v  concatvol -          ENABLED ACTIVE 102400 SELECT - fsgen
pl concatvol-01 concatvol ENABLED ACTIVE 102400 CONCAT - RW
sd disk03-01 concatvol-01 disk03 0 102400 0 hdisk3 ENA
# vxassist relayout concatvol01 layout=stripe ncol=2
# vxprint -tvh concatvol01
.....
v  concatvol01 -          ENABLED ACTIVE 102400 SELECT
concatvol01-01
fsgen
pl concatvol01-01 concatvol01 ENABLED ACTIVE 102400 STRIPE 2/128 RW
sd disk01-07 concatvol01-01 disk01 403400 51200 0/0 hdisk2 ENA
sd disk04-05 concatvol01-01 disk04 453600 51200 1/0 hdisk0 ENA
```

---

Example 5-23 shows the **vxassist** command being used to convert a mirrored volume to a layered RAID 1+0 volume.

*Example 5-23 Converting from RAID 0+1 to RAID 1+0*

---

```
# vxprint newvol01
Disk group: rootdg

TY NAME          ASSOC          KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
v  newvol01      fsgen          ENABLED 102400 -       ACTIVE -       -
pl newvol01-01  newvol01      ENABLED 102400 -       ACTIVE -       -
sd disk01-01    newvol01-01   ENABLED 51200  0       -       -       -
sd disk03-01    newvol01-01   ENABLED 51200  0       -       -       -
pl newvol01-03  newvol01      ENABLED 102400 -       ACTIVE -       -
sd disk02-03    newvol01-03   ENABLED 51200  0       -       -       -
sd disk04-03    newvol01-03   ENABLED 51200  0       -       -       -
# vxassist convert newvol01 layout=stripe-mirror
```

---

```
# vxprint newvol01
Disk group: rootdg
```

| TY | NAME         | ASSOC       | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|-------------|---------|--------|--------|--------|--------|--------|
| v  | newvol01     | fsgen       | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | newvol01-04  | newvol01    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sv | newvol01-S01 | newvol01-04 | ENABLED | 51200  | 0      | -      | -      | -      |
| sv | newvol01-S02 | newvol01-04 | ENABLED | 51200  | 0      | -      | -      | -      |

For a full list of supported layout and conversion types that are possible, refer to “Permitted relayout transformations” in Chapter 1, “Understanding VERITAS Volume Manager”, in the *VERITAS Volume Manager 3.2 - Administrators Guide (AIX)*.

Volume relayouts can take a long time to complete. The **vxtask** command or the **vxrelayout status** can be used to monitor the progress of a volume relayout. Example 5-24 shows the **vxtask** command and the **vxrelayout status** command being used to monitor the progress of a relayout from a concatenated mirror to a striped mirror.

#### Example 5-24 Monitoring a relayout

```
# vxtask list
TASKID PTID TYPE/STATE PCT PROGRESS
483 RELAYOUT/R 59.87% 0/409600/245224 RELAYOUT mirrvol01
# vxrelayout status mirrvol01
CONCAT-MIRROR --> STRIPED-MIRROR, columns=2, stwidth=128
Relayout running, 50.00% completed.
```

Volume layouts can be changed via the VEA by selecting the volume in the volume view, then selecting **Actions -> Change Layout**.

## 5.2.8 Renaming volumes

Volumes can be renamed with the **vxedit rename** command. The **vxedit** command can be used to rename the underlying objects of a volume, such as plexes and subdisks. Example 5-25 on page 151 shows a volume called vol01 being renamed to vol02. The following steps are performed:

1. The **vxprint** command is used to show the volume layout.
2. The volume is renamed to vol02 using the **vxedit rename** command.
3. The plex associated with the volume is renamed to vol02-01.
4. The **vxprint** command is run to show the new volume and plex names.

### Example 5-25 Renaming a volume

```
# vxprint vol01
Disk group: rootdg
```

| TY | NAME      | ASSOC    | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|-----------|----------|---------|--------|--------|--------|--------|--------|
| v  | vol01     | fsgen    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | vol01-01  | vol01    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk03-08 | vol01-01 | ENABLED | 102400 | 0      | -      | -      | -      |

```
# vxedit rename vol01 vol02
# vxedit rename vol01-01 vol02-01
# vxprint vol02
Disk group: rootdg
```

| TY | NAME      | ASSOC    | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|-----------|----------|---------|--------|--------|--------|--------|--------|
| v  | vol02     | fsgen    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | vol02-01  | vol02    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk03-08 | vol02-01 | ENABLED | 102400 | 0      | -      | -      | -      |

Volumes can be renamed via the VEA by selecting the volume in the volume view window and then selecting **Actions -> Rename Volume**.

To rename a volume using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Set Characteristics of a VxVM Volume -> Rename a VxVM Volume**, or use the `smitty vxvmlvrename` fast path.

## 5.2.9 Removing volumes

Before removing a volume, ensure that the data contained on the volume is either relocated or backed up if it is still required. The `vxedit` command is used for removing volumes.

Example 5-26 on page 152 shows an example of a volume with a file system called `/vrtfs1` mounted on it being removed. The following steps are performed:

1. The `df` command is run to find the volume.
2. The file system is unmounted.
3. The file system is removed using the `rmfs` command with the `-r` option, which will remove the mount point and remove the entry from the `/etc/filesystems` file.
4. The volume is stopped using the `vxvol stop` command.
5. The `vxedit` command is run with the `rm` option to remove the volume. The `-r` flag is used to recursively remove all associated plexes and subdisks within the volume.

*Example 5-26 Removing a volume*

```
# df -k /vrtfs1
Filesystem      1024-blocks      Free %Used      Iused %Iused Mounted on
/dev/vx/dsk/rootdg/raid5vol      51200      46937      9%          6      1% /vrtfs1
# umount /vrtfs1
# rmfs -r /vrtfs1
# vxvol stop raid5vol
# vxedit -rf rm raid5vol
```

To remove a volume using the VEA, select the volume in the **Volume View** window and then select **Actions -> Stop Volume**. Click **OK** on the confirmation message and then select **Actions -> Delete Volume**.

To remove a volume using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Remove a VxVM Volume**, or use the `smitty vxvmlvremove` fast path.

### 5.3 Disk group administration

This section describes some of tasks associated with disk group administration, such as adding a disk to a disk group, removing a disk group, and deporting and importing disk groups.

#### 5.3.1 Adding and removing disks from disk groups

To add a disk to a disk group, the `vx dg` command is used. To add a previously initialized disk known to the AIX operating system as `hdisk4` to disk group `vx dg01`, the following command is run:

```
# vx dg -g vx dg01 add disk disk04=hdisk4
```

The name that the disk is known to VxVM is specified as `disk04`. Example 5-27 shows the output of the `vx disk` command to show the disks in a disk group with the newly added `disk04`.

*Example 5-27 Listing disks in a disk group*

```
# vx disk -g vx dg01 list
DEVICE      TYPE      DISK      GROUP      STATUS
hdisk2      simple    disk02     vx dg01     online
hdisk3      simple    disk03     vx dg01     online
hdisk4      simple    disk04     vx dg01     online
```



To remove disk04 from disk group vx dg01, use the following command:

```
# vx dg -g vx dg01 rmdisk disk04
```

This disk will now be available to be used in another disk group.

To add a disk to a disk group using the VEA, select the disk group in the disk group view window, then select **Actions -> Add Disk to Dynamic Disk Group**.

To add a disk to a disk group using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Disk Administrator -> Add or Initialize disks**, or use the `smitty vxvmdiskadmadd` fast path. This is the same option for adding a disk, but it can also be used for adding a disk to a disk group.

### 5.3.2 Removing disk groups

If a disk group is no longer needed and there is no intention of moving it to another system, the `vx dg destroy` command can be used to remove the disk group.

**Attention:** Using `vx dg destroy` on a disk group will remove all volumes, plexes, and subdisks within a disk group. Ensure that all volumes are relocated if the data is required on these volumes before destroying a disk group.

To remove a disk group called vx dg01, run the following command:

```
# vx dg destroy vx dg01
```

The disks that were used in a disk group that has been removed are now available for use in other disk groups.

To remove a disk group via SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> Disk Groups -> Remove a Disk Group**, or the `smitty vxvmdgremove` fast path.

### 5.3.3 Deporting and importing disk groups

Disk groups can be moved between systems by first deporting the disk group from one system using the `vx dg deport` command and then importing the disk group on the destination system using the `vx dg import` command.

Example 5-28 shows an example of deporting a disk group called vxdg01 from a system called srvr50f. The following steps are performed:

1. The **vx dg list** command is run to list disk groups on the system.
2. All file systems mounted on the volumes are unmounted.
3. All volumes in the disk group are stopped by running the **vxvol stopall** command.
4. The **vxinfo** command is run to confirm the volumes are all stopped.
5. The disk group is deported using the **vx dg deport** command.
6. The **vx dg list** command is run to confirm the disk group no longer exists on the system.

*Example 5-28 Deporting a disk group*

---

```
# vx dg list
NAME          STATE      ID
rootdg        enabled    1029792565.1025.srvr50f
vx dg01        enabled    1030136439.1059.srvr50f

# umount /vrtfs
# vxvol -g vx dg01 stopall
# vxinfo -g vx dg01
mirrvol        fsgen      Startable
concatvol      fsgen      Startable
newvol         fsgen      Startable

# vx dg deport vx dg01
# vx dg list
NAME          STATE      ID
rootdg        enabled    1029792565.1025.srvr50f
```

---

The **vx disk -o all dgs list** command will list any disk that belongs to a disk group that has been deported. The deported disk group will be enclosed in braces for each disk, as shown in Example 5-29.

*Example 5-29 Listing deported disks*

---

| DEVICE | TYPE   | DISK   | GROUP       | STATUS |
|--------|--------|--------|-------------|--------|
| hdisk0 | simple | -      | -           | LVM    |
| hdisk1 | simple | -      | -           | LVM    |
| hdisk2 | simple | disk01 | rootdg      | online |
| hdisk3 | simple | -      | -           | LVM    |
| hdisk4 | simple | -      | (veritasdg) | online |
| hdisk5 | simple | -      | -           | online |
| hdisk6 | simple | -      | -           | LVM    |
| hdisk7 | simple | -      | -           | LVM    |

---

Example 5-30 shows the disk group deported in Example 5-28 on page 154 being imported onto a system called `srvr50a`. The following steps are performed:

1. After the disks have been physically moved to the new system or, if it is on a disk array attached to both systems, the **`vx dg import`** command is run. This will search all the disks on the system for the disk group information of the named disk group.
2. The **`vx dg list`** command is then run to show the newly imported disk group.
3. All volumes in the disk group are then started by using the **`vx vol startall`** command.
4. The **`vx info`** command is then run to confirm the volumes have been started.
5. The file systems can then be mounted and accessed on the new system.

*Example 5-30 Importing a disk group*

---

```
# vx dg import vx dg01
# vx dg list
NAME          STATE          ID
rootdg        enabled    1027624662.1025.srvr50a
vx dg01        enabled    1030136439.1059.srvr50a
# vx vol -g vx dg01 startall
# vx info -g vx dg01
mirrvol        fsgen        Started
concatvol      fsgen        Started
newvol         fsgen        Started
# mount -V vx fs /dev/vx/dsk/vx dg01/mirrvol /vx fs1
```

---

If a disk group is unable to be deported from a system due to a system failure, it can still be imported on the new system by specifying the **`-C`** flag with the **`vx dg import`** command. This will clear the import lock that is normally cleared during the deport process.

**Important:** Ensure that the disk group is not being used by another system when using the **`vx dg -C import`** command, as this could result in two different systems accessing the same disk group and cause data corruption.

To deport a disk group via the VEA, select the disk group from the disk group view and then select **Actions -> Deport Dynamic Disk Group**. To import a disk group, select the disk group and then select **Actions -> Import Dynamic Disk Group**.

To deport a disk group using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> Disk Groups -> Deport a Disk group**, or use the `smitty vxvmdgdeport` fast path. To import a disk group, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> Disk Groups -> Import a Disk Group**, or use the `smitty vxvmdgimport` fast path.

### ***Using deport and import to rename disk groups***

Deporting a disk group and then importing the disk group on the same or different system can be used for renaming a disk group if the `-n` flag is used with the `vxvg import` or `vxvg deport` command. For example, to rename a disk group called `vxvg01` to `vxvg02`, the following commands would be run:

```
# vxvg deport vxvg01
# vxvg -n vxvg02 import vxvg01
```

Alternatively, the disk group can be renamed on deport as follows:

```
# vxvg -n vxvg02 deport vxvg01
# vxvg import vxvg02
```

## **5.4 Backups and restores**

This section discusses some backup techniques using some features of VxFS and VxVM, such as:

- ▶ File system snapshots
- ▶ Volume snapshots
- ▶ Split mirror backups
- ▶ Backing up and restoring using `vxdump` and `vxrestore`

### **5.4.1 File system snapshots**

VxFS provides the ability to perform online snapshots of a mounted file system, which creates a point in time, read only copy of the source file system. In order to create a file system snapshot, a volume must exist that is large enough to hold a backup copy of the mounted file system. Also, the mount point that the snapshot file system is to be mounted on must exist. The volume that is to hold the backup copy of the VxFS file system can either be a VxVM volume or an AIX LVM volume.

The **mount** command is used to create a VxFS snapshot of an existing VxFS file system. To create a VxFS file system snapshot, perform the following steps:

1. Create either a VxVM or LVM volume large enough to hold the file system that is to be backed up.
2. Create a mount point that the backup copy of the file system is to be mounted on.
3. Use the **mount** command to create the snapshot by specifying the **-o snapof** option.

Example 5-31 shows an example of how to create a VxFS snapshot of a mounted file system called **/vrtfs** of 100 MB onto a VxVM volume called **backupvol**. The target volume is mounted on **/backup**. The target volume and the destination volume do not need to be of the same layout. The snapshot can then be backed up to tape.

*Example 5-31 Creating a VxFS file system snapshot*

---

```
# df -k /vrtfs
Filesystem      1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/vx/dsk/rootdg/mirrvol01      102400      43534   58%      239      3% /vrtfs
# mount -V vxfs -o snapof=/vrtfs /dev/vx/dsk/backupvol /backup
# df -k /backup
Filesystem      1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/vx/dsk/backupvol      102400      43522   58%      288      3% /backup
```

---

Once a snapshot of a file system is unmounted, the data on the backup file system is lost, but the volume will still exist and can be used for further snapshots.

File system snapshots can be performed via the VEA by selecting the source file system in the file system view window and then selecting **Actions -> Snapshot File System**.

## 5.4.2 Volume snapshots

Snapshots can be done at a volume level with VxVM volumes. This is achieved by a two step process, which does the following:

- ▶ Attaches a write only plex to the current volume and synchronizes the data on the write only plex with the data on the original plex.
- ▶ At a convenient time, the write only plex is detached from the volume and used to create a new volume.

In order to perform a snapshot backup, there must be enough disk space in the disk group to hold a copy of the volume being snapped. Use the **vx dg free** command to find out if there is enough space.

The **vxassist** command is used to create volume snapshots. To perform the first step, the **vxassist snapstart** command is used and for the second step the **vxassist snapshot** command is run.

Example 5-32 shows an example of the steps taken to perform a volume snapshot of volume that has a file system called /vrtfs mounted on it. The steps performed in this example are as follows:

1. Start the snapstart process by using the **vxassist** command. The **-b** option is used to run the process in the background.
2. The **vxtask** command is then used to view the progress of the resync. Alternatively, the **vxassist snapwait** command can be used to wait for the resync operation to complete, however this will run in the foreground until the snapstart completes. Below is an example of the **vxassist snapwait** command:  

```
# vxassist -g rootdg -b snapwait mirrvol01
Snapshot ready on volume mirrvol01
```
3. Once the snapstart has completed, the **vxprint** command is run to show the layout of the volume. Notice an extra plex has been created and is in the state of SNAPDONE.
4. The **vxassist snapshot** command is then run; this detaches the snapshot plex and uses it to create another volume. By default, the name of the new volume is SNAP-<volumename>, where <volumename> is the source volume name. To specify a different volume name, use the **-o name=** option with the **vxassist snapshot** command. For example, to create a snapped volume called snapvol01, run the following command:  

```
# vxassist -g rootdg snapshot mirrvol01 -o name=snapvol01
```
5. The **vxprint** command is then run to show the layout of the target snapped volume.
6. A file system check is then run on the newly created volume.
7. The newly created volume is then mounted on the mount point /backup.
8. Finally, the **df** command is run on the target and source file systems to show a comparison.

*Example 5-32 Creating a volume snapshot*

---

```
# vxassist -g rootdg -b snapstart mirrvol01
# vxtask list
TASKID PTID TYPE/STATE PCT PROGRESS
```

```

451          ATCOPY/R 43.47% 0/204800/89032 PLXATT mirrvol01 mirrvol01-03
# vxprint mirrvol01
Disk group: rootdg

TY NAME      ASSOC      KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
v mirrvol01  fsgen      ENABLED 204800  -       ACTIVE -       -
pl mirrvol01-01 mirrvol01  ENABLED 204800  -       ACTIVE -       -
sd disk02-01 mirrvol01-01 ENABLED 204800  0       -       -       -
pl mirrvol01-02 mirrvol01  ENABLED 204800  -       ACTIVE -       -
sd disk03-02 mirrvol01-02 ENABLED 204800  0       -       -       -
pl mirrvol01-03 mirrvol01  ENABLED 204800  -       SNAPDONE -       -
sd disk04-02 mirrvol01-03 ENABLED 204800  0       -       -       -
# vxassist -g rootdg snapshot mirrvol01
# vxprint SNAP-mirrvol01
Disk group: rootdg

TY NAME      ASSOC      KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
v SNAP-mirrvol01 fsgen      ENABLED 204800  -       ACTIVE -       -
pl mirrvol01-03 SNAP-mirrvol01 ENABLED 204800  -       ACTIVE -       -
sd disk04-02 mirrvol01-03 ENABLED 204800  0       -       -       -
# fsck -y -V vxfs /dev/vx/dsk/rootdg/SNAP-mirrvol01
log replay in progress
replay complete - marking super-block as CLEAN
# mount -V vxfs /dev/vx/dsk/rootdg/SNAP-mirrvol01 /backup
# df -k /vrtfs
Filesystem    1024-blocks    Free %Used    Iused %Iused Mounted on
/dev/vx/dsk/rootdg/mirrvol01    102400    43534 58%    240    3% /vrtfs
# df -k /backup
Filesystem    1024-blocks    Free %Used    Iused %Iused Mounted on
/dev/vx/dsk/rootdg/SNAP-mirrvol01    102400    43534 58%    240    3%
/backup

```

Once the snapshot has been completed it can be backed up to tape. Once the snapshot has been backed up to tape, there are three options available on what can be done with the snapshot plex:

- It can be reattached to the volume to get another snapshot by using the **vxassist snapback** command. For example, to reattach the snapshot plex created in Example 5-32 on page 158, run the following command:

```
# vxassist snapback SNAP-mirrvol01
```

This will reattach and synchronize the snapshot plex with the original volume and can be used for another backup.

**Note:** The snapback feature is not supported on RAID 5 volumes.

- It can be disassociated totally from the volume for use elsewhere by using the **vxassist snapclear** command. For example, to totally disassociate the snapped volume created in Example 5-32 on page 158 from its source volume, run the following command:

```
# vxassist snapclear SNAP-mirrvo101
```

- It can be deleted to free up the disk space by using the **vxedit -rf rm** command. For example, to totally delete the snapshot created in Example 5-32 on page 158, run the following command:

```
# vxedit -rf rm SNAP-mirrvo101
```

Volume snapshots can be performed via the VEA by selecting the volume in the volume view window, then selecting **Actions -> Snap -> Snap Start**. When the snapshot is ready to be taken, select **Actions -> Snap -> Snap Shot**.

To perform a snapshot using SMIT, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Volumes -> Snapshot of a Volume for Back Up**, or use the **smitty vxvmlvsnapshot** fast path.

### 5.4.3 Split mirror backups

Existing mirrored volumes can be backed up to tape by disassociating one of the plexes from the volume and using that plex to form another volume, which would be a point in time copy of the original volume. The new volume can then be backed up to tape. When the backup has completed, the plex on the new volume can then be disassociated from the volume and reattached to the original volume. Example 5-33 on page 161 shows an example of creating a volume from a plex of a volume called **mirrvo101** that is mounted on **/vrtfs**. The steps performed in this example are as follows:

1. The **vxprint** command is run to show the existing layout of the volume.
2. The plex **mirrvo101-02** is disassociated from the volume.
3. The plex is then used to create another volume called **backupvol** by using the **vxmake** command.
4. The volume is then started using the **vxvol start** command.
5. The **vxprint** command is then run to show the layout of the new volume.
6. A file system check is then run on the volume by using the **fsck** command.
7. The new volume is then mounted on the mount point **/backup**.
8. Finally, the **df** command is used to compare the original file system and the new file system. Note that this may differ as updates will continue to occur on the original file system while the new volume is in the process of being



created. The newly created file system or volume can now be backed up to tape.

*Example 5-33 Creating a volume from the plex of a mirrored volume*

```
# vxprint -tvh mirrvol01
Disk group: rootdg

V  NAME          RVG          KSTATE  STATE    LENGTH  READPOL  PREFPLEX
UTYPE
PL NAME          VOLUME        KSTATE  STATE    LENGTH  LAYOUT   NCOL/WID  MODE
SD NAME          PLEX          DISK    DISKOFFS LENGTH  [COL/]OFF  DEVICE    MODE
SV NAME          PLEX          VOLNAME  NVOLLAYR LENGTH  [COL/]OFF  AM/NM     MODE
DC NAME          PARENTVOL     LOGVOL
SP NAME          SNAPVOL       DCO

v  mirrvol01      -              ENABLED  ACTIVE   204800   SELECT   -         fsgen
pl mirrvol01-01  mirrvol01     ENABLED  ACTIVE   204800   CONCAT   -         RW
sd disk02-01     mirrvol01-01 disk02    0        204800   0        hdisk4    ENA
pl mirrvol01-02 mirrvol01     ENABLED  ACTIVE   204800   CONCAT   -         RW
sd disk03-02     mirrvol01-02 disk03    102400   204800   0        hdisk3    ENA
# vxplex -g rootdg dis mirrvol01-02
# vxmake -Ufsgen vol backupvol plex=mirrvol01-02
# vxvol start backupvol
# vxprint -tvh backupvol
Disk group: rootdg

V  NAME          RVG          KSTATE  STATE    LENGTH  READPOL  PREFPLEX
UTYPE
PL NAME          VOLUME        KSTATE  STATE    LENGTH  LAYOUT   NCOL/WID  MODE
SD NAME          PLEX          DISK    DISKOFFS LENGTH  [COL/]OFF  DEVICE    MODE
SV NAME          PLEX          VOLNAME  NVOLLAYR LENGTH  [COL/]OFF  AM/NM     MODE
DC NAME          PARENTVOL     LOGVOL
SP NAME          SNAPVOL       DCO

v  backupvol      -              ENABLED  ACTIVE   204800   ROUND    -         fsgen
pl mirrvol01-02  backupvol     ENABLED  ACTIVE   204800   CONCAT   -         RW
sd disk03-02     mirrvol01-02 disk03    102400   204800   0        hdisk3    ENA

# fsck -y -V vxfs /dev/vx/dsk/rootdg/backupvol
log replay in progress
replay complete - marking super-block as CLEAN
# mount -V vxfs /dev/vx/dsk/rootdg/backupvol /backup
# df -k /vrtfs
Filesystem    1024-blocks      Free %Used      Iused %Iused Mounted on
/dev/vx/dsk/rootdg/mirrvol01      102400      43534  58%      240      3% /vrtfs
# df -k /backup
Filesystem    1024-blocks      Free %Used      Iused %Iused Mounted on
```

|                              |        |       |     |     |    |
|------------------------------|--------|-------|-----|-----|----|
| /dev/vx/dsk/rootdg/backupvol | 102400 | 43534 | 58% | 240 | 3% |
| /backup                      |        |       |     |     |    |

Once the newly created volume or file system has been backed up to tape, the plex used on the new volume can be disassociated with the new volume and reattached to the original volume. Example 5-34 shows how to reattach the plex used for the backup volume back to its original volume. The steps performed here are:

1. The file system used for the backup is unmounted.
2. The **vxprint** command is used to show the backup volume's layout.
3. The **vxplex** command is used to disassociate the plex from the volume. Since the volume only contains one plex, the -f option is used to force the process. The following message will be displayed if the -f option is not used:  
  

```
vxvm:vxplex: ERROR: Plex mirrvol01-02, volume backupvol: Detaching last plex requires use of -f
```
4. The plex used in the backup volume is reattached to the original volume. It may take some time to resynchronize the data and the progress can be monitored by using the **vxtask** command.
5. The **vxprint** command is used to confirm that the layout of the original volume is back to how it was before the backup.
6. Finally, the **vxedit** command is used to remove the backup volume.

*Example 5-34 Reattaching plex to original volume*

```
# umount /backup
# vxprint backupvol
Disk group: rootdg
```

| TY | NAME         | ASSOC        | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|--------------|---------|--------|--------|--------|--------|--------|
| v  | backupvol    | fsgen        | ENABLED | 204800 | -      | ACTIVE | -      | -      |
| p1 | mirrvol01-02 | backupvol    | ENABLED | 204800 | -      | ACTIVE | -      | -      |
| sd | disk03-02    | mirrvol01-02 | ENABLED | 204800 | 0      | -      | -      | -      |

```
# vxplex -g rootdg -f dis mirrvol01-02
# vxplex -g rootdg att mirrvol01 mirrvol01-02
# vxprint mirrvol01
Disk group: rootdg
```

| TY | NAME         | ASSOC        | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|--------------|---------|--------|--------|--------|--------|--------|
| v  | mirrvol01    | fsgen        | ENABLED | 204800 | -      | ACTIVE | -      | -      |
| p1 | mirrvol01-01 | mirrvol01    | ENABLED | 204800 | -      | ACTIVE | -      | -      |
| sd | disk02-01    | mirrvol01-01 | ENABLED | 204800 | 0      | -      | -      | -      |
| p1 | mirrvol01-02 | mirrvol01    | ENABLED | 204800 | -      | ACTIVE | -      | -      |
| sd | disk03-02    | mirrvol01-02 | ENABLED | 204800 | 0      | -      | -      | -      |

## 5.4.4 Using vxdump and vxrestore

There are two commands provided by VxFS for backup and restore to tape of a VERITAS file system: they are **vxdump** for backups and **vxrestore** for restores of backups created by **vxdump**.

### Backing up using vxdump

The **vxdump** command can be used to back up a VxFS to tape or to a file. The **vxdump** command supports incremental backups. Example 5-35 shows an example of backing up a file system called **/vrtfs** to a tape drive using **vxdump**. This example shows a full backup, also known as a level 0 backup, specified with the **-0** option. The **u** option specifies updating the **/etc/dumpdates** file, which keeps a record of all backups performed by **vxdump**. The **/etc/dumpdates** file is also used by AIX's **backup** command. The two backup commands, **vxdump** and **backup**, can use the same **/etc/dumpdates** file without one overwriting information written to by the other command. To keep **vxdump** and **backup** from using the same **/etc/dumpdates** file, the **-o** option can be used with **vxdump**, which will cause **vxdump** to use the **/etc/vxdumpdates** file.

*Example 5-35 Using vxdump to back up a VxFS file system*

---

```
# vxdump -0uf /dev/rmt0 /vrtfs
vxfs vxdump: Date of this level 0 dump: Fri Aug 23 10:30:47 2002
vxfs vxdump: Date of last level 0 dump: the epoch
vxfs vxdump: Dumping /dev/vx/rdsk/rootdg/mirrvol01 (/vrtfs) to /dev/rmt0
vxfs vxdump: mapping (Pass I) [regular files]
vxfs vxdump: mapping (Pass II) [directories]
vxfs vxdump: estimated 110106 blocks (53.76MB).
vxfs vxdump: dumping (Pass III) [directories]
vxfs vxdump: dumping (Pass IV) [regular files]
vxfs vxdump: vxdump: 55134 tape blocks on 1 volumes(s)
vxfs vxdump: level 0 dump on Fri Aug 23 10:30:47 2002
vxfs vxdump: Closing /dev/rmt0
vxfs vxdump: vxdump is done
```

---

Example 5-36 on page 164 shows an example of the **/etc/dumpdates** file. It has three columns:

- ▶ The raw device of the file system that was backed up.
- ▶ The dump level: This is a number from 1 to 9. For a description of dump levels, see “Dump levels” on page 164.
- ▶ The date the backup was performed.

Example 5-36 shows an `/etc/dumpdates` file, which contains three VxFS file system backups (two full backups and a level 1 incremental backup), and a full backup used by the **backup** command to back up a file system on an LVM logical volume.

*Example 5-36 Example of the `/etc/dumpdates` file*

---

```
# cat /etc/dumpdates
/dev/vx/rdisk/rootdg/mirrvol01 0 Fri Aug 23 10:30:47 2002
/dev/vx/rdisk/rootdg/raid5vol 0 Fri Aug 23 10:24:23 2002
/dev/vx/rdisk/rootdg/mirrvol01 1 Fri Aug 23 10:34:48 2002
/dev/rhd10opt 0 Fri Aug 23 10:37:08 2002
```

---

### ***Dump levels***

There are nine dump levels that can be specified with the **vxdump** command. A level 0 dump will back up all files in a file system. A level 1 dump will back up all files modified since the last level 0 dump. A level 9 dump will back up all files modified since the next lower level found for the same file system in the `/etc/dumpdates` file. For example, if a file system has level 1, 2, and 4 dump records in the `/etc/dumpdates` file and a level 9 dump is performed, then all files modified since the level 4 dump was performed will be backed up.

### **Restoring using vxrestore**

The **vxrestore** command is used to restore file systems or individual files that were backed up by the **vxdump** command. To restore the entire contents of a file system from backup tapes, perform the following steps:

1. Recreate the VxVM volume if necessary.
2. Create and mount a new file system on the volume.
3. Restore the level 0 backup from tape.
4. Restore other incremental levels from the lowest to highest level. For example, restore from a level 3 backup before restoring from a level 4 backup.

Example 5-37 on page 165 shows an example of using **vxrestore** to restore the contents of the file system backed up in Example 5-35 on page 163. The following steps are performed:

1. The file system is recreated using the **crfs** command.
2. The file system is mounted using the **mount** command.
3. The current directory is changed to the mount point.
4. **vxrestore** is run to extract the files from the backup tape.

### Example 5-37 Restoring a file system using vxrestore

---

```
# /sbin/helpers/vxfs/crfs -v vxfs -d mirrvol01 -g rootdg -T "vv" -m /vrtfs
# mount /vrtfs
# cd /vrtfs
# vxrestore -rvf /dev/rmt0
vxfs vxrestore: warning: ./lost+found: File exists
Verify tape and initialize maps
Tape block size is 63
Dump      date: Fri Aug 23 11:21:38 2002
Dumped from: the epoch
level 0 dump of /vrtfs on /dev/vx/rdisk/rootdg/mirrvol0
Begin level 0 restore
Initialize symbol table.
  Extract directories from tape
Calculate extraction list.
Make node ./jre
.....
extract file ./jre/bin/java
extract file ./jre/bin/javaw
extract file ./jre/bin/jvmtcf
extract file ./jre/bin/keytool
extract file ./jre/bin/libJdbcOdbc.a
extract file ./jre/bin/libagent.a
extract file ./jre/bin/libawt.a
.....
extract file ./file
Add links
  Set directory mode, owner, and times.
Check the symbol table.
Check pointing the restore
```

---

### Restoring individual files using vxrestore

Files can be individually restored using **vxrestore** without having to restore the entire file system by using the **-i** option. The **-i** option causes **vxrestore** to enter an interactive mode where files can be selected for restore.

Example 5-38 on page 166 shows an example of how to use **vxrestore** to restore the **jre.tar** file, which was backed up by the **vxdump** command in Example 5-35 on page 163. The steps performed in this example are:

1. The current directory is changed to **/vrtfs**.
2. **vxrestore** is run, specifying the **-i** option; **vxrestore** then enters an interactive prompt.
3. From the interactive prompt, the **ls** command is run to list the files and sub directories contained in the backup.

4. The **add** command is run followed by the file name to add the file to the list of file names to be restored.
5. The **ls** command is run again to list the files. Note that the `jre.tar` file has an asterisk next to it, indicating it has been selected for restore.
6. Finally, the **extract** command is run to extract the file from the backup. The **vxrestore** interactive prompt asks for the volume to restore from, the number corresponding to the tape is entered. **vxrestore** then prompts to set the original mode of the file before it was backed up. The interactive mode is then exited by entering `quit`.

---

*Example 5-38 Using vxrestore to restore individual files*

---

```
# cd /vrtfs
# vxrestore -if /dev/rmt0
vxrestore > ls
.:
file          jre/          jre.tar        lost+found/

vxrestore > add jre.tar
vxrestore > ls
.:
file          jre/          *jre.tar       lost+found/

vxrestore > extract
Specify next volume #: 1
set owner/mode for '.'? [yn] y
vxrestore > quit
```

---

From the **vxrestore** interactive prompt, wildcards such as `*` can be used to select multiple files. If a directory is selected, the entire contents of the directory will be restored. Files can be viewed in sub directories by changing to the sub directory by using the `cd` command then by using the `ls` command.

## 5.5 Problem prevention and resolution

This section discusses ways that problems can be avoided that will cause potential loss of data contained on VxFS file systems and VxVM volumes. The following topics are contained in this section:

- ▶ Hot relocation
- ▶ Hot sparing
- ▶ Evacuating disks
- ▶ Replacing failed disks

## 5.5.1 Hot relocation

During system startup, the **vxrelocd** daemon is started by default. This process monitors for any volume, plex, and subdisk failures and will relocate any subdisks that reside on a disk that is failing to other subdisks that are not on the failing disk. Hot relocation will only relocate subdisks that are part of a redundant volume; failed subdisks on simple concatenated or RAID 0 volumes will not be relocated by the **vxrelocd** daemon; however, an e-mail message will be sent to the root user ID notifying of a failure of a subdisk on all types of volumes. The following list determines the circumstances on which space is selected for hot relocation:

- ▶ Any disk that has been pre-designated as a hot spare.
- ▶ If a hot spare has not been designated, then any free space that is available in the volume will be used to relocate the subdisks unless the redundancy of the volume cannot be maintained.

To mark a disk as a hot spare in a volume group, use the **vxedit** command, as shown in Example 5-39. The **vxdg spare** command can be used to list disks that are designated as hot spare disks, as also shown in Example 5-39. Disks can also be marked as hot spares by selecting option 11 “Mark a disk as a spare for a disk group” from **vxdiskadm**.

*Example 5-39 Marking a disk as a hot spare*

---

```
# vxedit set spare=on disk03
# vxdg -g vxdg01 spare
```

| DISK   | DEVICE | TAG    | OFFSET | LENGTH   | FLAGS |
|--------|--------|--------|--------|----------|-------|
| disk03 | hdisk3 | hdisk3 | 0      | 17771728 | s     |

---

If a disk is designated as a hot spare, it cannot be used for creating other VxVM objects. Other disks that are in a disk group can also be excluded from use by hot relocation by using the **vxedit** command. For example, to set disk04 in disk group vxdg01 to not be used for hot relocation, run the following command:

```
# vxedit set nohotuse=on disk04
```

Alternatively a disk can be excluded from use by hot relocation by selecting option 14 “Exclude a disk from hot-relocation use” from the **vxdiskadm** main menu.

Example 5-40 shows how the hot spare and hot relocation status of each disk in a disk group can be displayed by using the **vxdisk** command.

*Example 5-40 Listing disk hot relocation status of disk*

---

```
# vxdisk -g vxdg01 list
```

| DEVICE | TYPE | DISK | GROUP | STATUS |
|--------|------|------|-------|--------|
|--------|------|------|-------|--------|

---

|        |        |        |         |                 |
|--------|--------|--------|---------|-----------------|
| hdisk3 | simple | disk03 | vx dg01 | online spare    |
| hdisk4 | simple | disk04 | vx dg01 | online nohotuse |

---

Once a failure has occurred and hot relocation has taken place, the root user will receive an e-mail message warning of the failure. If hot relocation has failed, the e-mail message is still sent to the root user and the problem will need to be rectified manually. After a successful relocation has taken place, it may still be necessary for the administrator to change the layout of a volume. The user ID that the e-mail message is sent to can be changed by editing the `/etc/init.d/vxvm-recover` script and changing the following line:

```
vxrelocd root &
```

Change the root user to another user ID or, alternatively, list other user IDs after the root user ID.

Disks can be marked as hot spares or excluded from hot relocation via the VEA by selecting the disk in the disk view and then selecting **Actions -> Set Disk Usage**. A window then appears with three options: **Mark disk hot relocation or hot spare**, **Reserve disk**, and **Do not use disk for hot relocation or hot spare**. Click the check box for the required option and then click on **OK**.

## 5.5.2 Hot sparing

Hot sparing can be enabled in favor of hot relocation by editing the `/etc/init.d/vxvm-recover` script and commenting out the following line:

```
vxrelocd root &
```

and uncommenting the following line:

```
#vxsparecheck root &
```

On the next system reboot, hot sparing will be enabled instead of hot relocation.

**Important:** Hot relocation and hot sparing cannot coexist; `vxsparecheck` and `vxrelocd` cannot be running at the same time.

Hot sparing enables more control over where a disk will be relocated to in the event of a disk failure. In the event of a disk failure, the criteria for disks to fail over to are as follows:

- ▶ **vxsparecheck** checks the `/etc/vx/sparelist` file for an appropriate disk.
- ▶ If a disk is not found in the `/etc/vx/sparelist` file, or the file does not exist, then a disk is selected in the disk group that is marked as a hot spare.



Unlike hot relocation, hot sparing will not relocate subdisks to disks that are not designated as hot spares. Example 5-41 shows an example of the format of the `/etc/vx/sparelist` file. Each line consists of a disk group name, a disk in the disk group that is used for containing volumes, and a list of disk(s) that are designated hot spares, which will be tried in order should the data disk fail.

*Example 5-41 /etc/vx/sparelist file*

---

```
# cat /etc/vx/sparelist
vxdg01 disk03 : disk04
vxdg02 disk05 : disk09 disk10
```

---

**Important:** The `/etc/vx/sparelist` file is not checked for correctness and is not used until a disk has failed. It is the administrators responsibility to ensure this file is accurate.

### 5.5.3 Evacuating volumes from a disk

If a disk is failing or if a disk needs to be removed from the system for any reason, the volumes that occupy the disk can be moved to another disk by using either **`vxdiskadm`** or **`vxevac`**. This procedure will work for all volume layouts and can be performed without stopping I/O to the volume; however, there are some situations where it will fail, such as:

- ▶ Not enough space available on the destination disks in the disk group.
- ▶ For mirrored volumes, a subdisk in a plex is attempted to be moved onto another disk that already contains subdisks that make up one of the other plexes of the same volume.
- ▶ For RAID 5 volumes, a disk is attempted to be moved onto another disk that already has subdisks used as part of the RAID 5 volume.

Example 5-42 on page 170 shows an example of using **`vxevac`** to move all the volumes in disk group `vxdg01` on disk `disk04` to `disk03`. The following steps are performed:

1. The **`vxdisk list`** command is run to show the disks in the disk group.
2. The layout of two volumes that occupy `disk04` is shown using the **`vxprint`** command.
3. The **`vxevac`** command is run to move all volumes from `disk04` to `disk03`. This process may take a while and can be monitored by the **`vxtask`** command. Multiple disks can be specified on the destination list. For example, to move volumes on `disk04` to `disk03` and `disk05` if the volume group had a disk called `disk05`, the following command would be run:

```
# vxevac -g vxdg01 disk04 disk03 disk05
```

- Finally, the **vxprint** command is run to show the layout of the volumes has been moved to the new disk.

*Example 5-42 Evacuating volumes from a disk using vxevac*

```
# vxdisk -g vxdg01 list
```

| DEVICE | TYPE   | DISK   | GROUP  | STATUS |
|--------|--------|--------|--------|--------|
| hdisk2 | simple | disk04 | vxdg01 | online |
| hdisk3 | simple | disk03 | vxdg01 | online |
| hdisk4 | simple | disk02 | vxdg01 | online |

```
# vxprint concatvol
Disk group: vxdg01
```

| TY | NAME         | ASSOC        | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|--------------|---------|--------|--------|--------|--------|--------|
| v  | concatvol    | fsgen        | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | concatvol-01 | concatvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk04-01    | concatvol-01 | ENABLED | 102400 | 0      | -      | -      | -      |

```
# vxprint mirrvol
Disk group: vxdg01
```

| TY | NAME       | ASSOC      | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|------------|------------|---------|--------|--------|--------|--------|--------|
| v  | mirrvol    | fsgen      | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | mirrvol-01 | mirrvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk02-01  | mirrvol-01 | ENABLED | 102400 | 0      | -      | -      | -      |
| pl | mirrvol-02 | mirrvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk04-02  | mirrvol-02 | ENABLED | 102400 | 0      | -      | -      | -      |

```
# vxevac -g vxdg01 disk04 disk03
# vxprint concatvol
Disk group: vxdg01
```

| TY | NAME         | ASSOC        | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|--------------|--------------|---------|--------|--------|--------|--------|--------|
| v  | concatvol    | fsgen        | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | concatvol-01 | concatvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk03-01    | concatvol-01 | ENABLED | 102400 | 0      | -      | -      | -      |

```
# vxprint mirrvol
Disk group: vxdg01
```

| TY | NAME       | ASSOC      | KSTATE  | LENGTH | PLOFFS | STATE  | TUTILO | PUTILO |
|----|------------|------------|---------|--------|--------|--------|--------|--------|
| v  | mirrvol    | fsgen      | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| pl | mirrvol-01 | mirrvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk02-01  | mirrvol-01 | ENABLED | 102400 | 0      | -      | -      | -      |
| pl | mirrvol-02 | mirrvol    | ENABLED | 102400 | -      | ACTIVE | -      | -      |
| sd | disk03-02  | mirrvol-02 | ENABLED | 102400 | 0      | -      | -      | -      |

The steps required to perform the same process using **vxdiskadm** are as follows:

1. Run **vxdiskadm** and select option 6 “Move volumes from a disk”. A description of the menu option appears followed by the following message:

Enter disk name [<disk>,list,q,?]

2. Type in **list**; a list of all disks on the system grouped by disk group is listed. Select the disk to be evacuated by typing in the name of the disk, as specified in the name column, and pressing Enter.

3. A message appears describing that the disk can be moved onto more than one destination disk by specifying a list of disks separated by spaces or, if left blank, then any available space will be used. The following prompt is displayed:

Enter disks [<disk ...>,list,q,?]

4. Type in **list**; a list of disks is displayed in the disk group. Type in the disk or a list of disks separated by spaces, then press Enter; the following warning is displayed:

Requested operation is to move all volumes from disk disk04 in group vxdg01.

NOTE: This operation can take a long time to complete.

Continue with operation? [y,n,q,?] (default: y)

5. Press Enter; the disk evacuation starts. As each volume is evacuated from the disk, it is listed as follows:

Move volume concatvol ...  
Move volume mirrvol ...  
Evacuation of disk disk04 is complete.

6. **vxdiskadm** then asks whether to evacuate more disks. Select the default of no by pressing Enter. The main menu is then displayed; press q and then Enter to exit **vxdiskadm**.

Disks can be evacuated via the VEA by selecting the disk in the disk view and then selecting **Actions -> Evacuate Disk**.

## 5.5.4 Replacing or removing disks

Failed disks or partially failing disks can be seen in the output of the **vxdisk list** command. Example 5-43 on page 173 shows an example of a disk that is partially failing. If a disk is failing and it contains volumes that are not in a mirrored or RAID 5 layout, then an attempt should be made to evacuate these volumes, as the data on these volumes will be lost as part of the disk removal process. See 5.5.3, “Evacuating volumes from a disk” on page 169 for a

description of how to evacuate volumes from a disk. If the volume is mirrored or in a RAID 5 layout, then the disk replacement procedure will recreate the layout after the disk is replaced.

To replace a disk that has partially failed, there are two steps:

1. Tag the disk as being removed
2. Remove the disk

To replace a disk that is failing, perform the following steps:

1. Evacuate the disk onto another disk if it contains simple concatenated volumes or unmirrored stripe volumes. See 5.5.3, “Evacuating volumes from a disk” on page 169 for a description of how to do this. Evacuation is not necessary if the disk contains mirrored or RAID 5 volumes.
2. Run the **vxdiskadm** command and select option 3 “Remove a disk for replacement”, A description of what the option will do is displayed followed by the following message:

```
Enter disk name [<disk>,list,q,?]
```

3. Type in **list**; a list of disks by disk group is then displayed, in which the failing disk can be identified. Type in the disk name of the disk that is failing. The following message is displayed:

```
The following volumes will lose mirrors as a result of this
operation:
```

```
mirrvol
```

```
No data on these volumes will be lost.
```

```
The requested operation is to remove disk disk04 from disk group
vxdg01. The disk name will be kept, along with any volumes using
the disk, allowing replacement of the disk.
```

```
Select "Replace a failed or removed disk" from the main menu
when you wish to replace the disk.
```

```
Continue with operation? [y,n,q,?] (default: y)
```

4. Press Enter; a message indicating the removal of the disk has been completed will be displayed, followed by a prompt to remove another disk. Select the default of not removing another disk by pressing Enter. The main menu is then redisplayed.
5. The disk is now put into a removed state; the **vxdisk list** command shows this disk in a removed state, as shown in Example 5-45 on page 174. When the disk has been physically replaced, run the **vxdiskadm** command again and select option 4 “Replace a failed or removed disk”.

6. A screen describing what the option will do followed by the following prompt will appear:  
Select a removed or failed disk [<disk>,list,q,?]
7. Type in `list` and then press Enter; a list of disks that are tagged as removed or failed will be displayed. Type in the disk name of the disk to be replaced and then press Enter.
8. If there are other unused disks available, a list will be displayed, and an option of using one of these disks as the replacement will displayed; otherwise, the following option will be displayed:  
Select disk device to initialize [<address>,list,q,?]
9. Type in `list` and then press Enter; a list of disks will be displayed. Type in the name of the newly replaced disk and then press Enter. The following message is displayed:  
The requested operation is to initialize disk device `hdisk2` and to then use that device to replace the removed or failed disk `disk04` in disk group `vxdg01`.  
Continue with operation? [y,n,q,?] (default: y)
10. Take the default option by pressing Enter. The following message is displayed:  
Use a default private region length for the disk?  
[y,n,q,?] (default: y)
11. Take the default option by pressing Enter; the disk is then replaced and a message similar to the following is then displayed:  
Replacement of disk `disk04` in group `vxdg01` with disk device `hdisk2` completed successfully.  
Replace another disk? [y,n,q,?] (default: n)
12. Take the default option of not replacing another disk by pressing Enter; the **`vxdiskadm`** main menu is then displayed. Press `q` and then Enter to exit **`vxdiskadm`**.
13. After the disk is replaced, it may take some time to resynchronize any mirrors or RAID 5 parity checks. Use the **`vxtask`** command to view the progress of these resynchronizations.

*Example 5-43 Partial disk failure*

---

```
# vxdisk list
DEVICE    TYPE    DISK    GROUP    STATUS
hdisk0    simple  -       -        LVM
hdisk1    simple  diska   rootdg   online
hdisk2    simple  -       -        online
hdisk3    simple  disk03  vxdg01   online
```

|        |        |        |         |                |
|--------|--------|--------|---------|----------------|
| hdisk4 | simple | disk04 | vx dg01 | online failing |
|--------|--------|--------|---------|----------------|

For a complete disk failure, it may not be possible to mark a disk as being removed via option 3 “Remove a disk for replacement”. In this case, the **vx disk list** output shown in Example 5-44 will show the disk as failed. If the disk has completely failed, it is only necessary to replace the disk and then select option 4 “Replace a failed or removed disk”.

Example 5-44 Complete disk failure

| # vx disk list |        |        |         |                   |
|----------------|--------|--------|---------|-------------------|
| DEVICE         | TYPE   | DISK   | GROUP   | STATUS            |
| hdisk0         | simple | -      | -       | LVM               |
| hdisk1         | simple | diska  | rootdg  | online            |
| hdisk2         | simple | -      | -       | online            |
| hdisk3         | simple | disk03 | vx dg01 | online            |
| hdisk4         | simple | -      | -       | online            |
| -              | -      | disk04 | vx dg01 | failed was:hdisk2 |

Example 5-45 Example of output from a removed disk

| # vx disk list |        |        |         |                    |
|----------------|--------|--------|---------|--------------------|
| DEVICE         | TYPE   | DISK   | GROUP   | STATUS             |
| hdisk0         | simple | -      | -       | LVM                |
| hdisk1         | simple | diska  | rootdg  | online             |
| hdisk2         | simple | -      | -       | online             |
| hdisk3         | simple | disk03 | vx dg01 | online             |
| hdisk4         | simple | disk02 | vx dg01 | online             |
| -              | -      | disk04 | vx dg01 | removed was:hdisk2 |

In some cases, disk failures may be caused by problems other than the actual disk itself, for example, a controller problem. In this case, the replacing a failed disk procedure can be followed without actually physically replacing the disk, or the **vxreattach** command can be run. If a controller problem has been fixed and the disk is not multipathed, the **vxreattach** command can be run as follows to bring the disk back into the configuration:

```
# vxreattach -br
```

This will locate any disks that have become available again and attempt to reattach them. The -b option will run the command in the background and the -r option will repair any plexes on the volumes.

Disks can be replaced via the VEA by selecting the disk from the disk view and then selecting **Actions -> Replace Disk**.

SMIT disks can be marked for removal by selecting **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Disk Administrator -> Remove a disk for replacement**. When the disk is ready to be replaced, select **System Storage Management (Physical & Logical Storage) -> VERITAS Volume Manager -> VxVM Disk Administrator -> Replace a failed or removed disk**.

## 5.6 File system administration

This section discusses tasks associated with VxFS file systems, such as:

- ▶ Determining block and extent log size
- ▶ Setting quotas
- ▶ Defragmenting file systems
- ▶ Optionally licensable features

### 5.6.1 Setting block and intent log size

The block and intent log size are set when the file system has been created and cannot be changed. The block size may need to be changed from the default of 1 KB depending on the application being run on the file system. For a file system with many small files, the default block size of 1 KB should be adequate. For file systems with larger and fewer files, a larger block size may be needed. To set the block size, use `-o bsize` with the `mkfs` command or `-absize` with the `crfs` command. For example to create a file system on volume `newvol01` in disk group `vxdg01` with a block size of 8 KB, run the following command:

```
mkfs -V vxfs -o bsize=8192 /dev/vx/rdisk/vxdg01/newvol01
```

or

```
/sbin/helpers/vxfs/vxfs/crfs -v vxfs -d newvol01 -g vxdg01 -Tvv -absize=8192
```

To set the intent log size, `-o logsize` is used with the `mkfs` command and `-a logsize` is used with the `crfs` command. The default size of 1024 blocks is usually sufficient. To create a file system on a volume called `newvol01` with a log size of 2048 blocks, run the following command:

```
mkfs -V vxfs -o logsize=2048 /dev/vx/rdisk/vxdg01/newvol01
```

or

```
/sbin/helpers/vxfs/vxfs/crfs -v vxfs -d newvol01 -g vxdg01 -Tvv -a logsize=2048
```

## 5.6.2 Quotas

Quotas can be used on VxFS file systems to control how much space a particular user or group has allocated for use on a particular file system. The commands used for quota allocation differ from the standard AIX quota administration commands.

### Enabling quotas

To turn on quotas for a file system, make sure there is a file called `quotas` and `quotas.grp` in the root directory of the file system (if it is already mounted). If these files do not exist use the **touch** command to create them. Once these files are created, use the **vxquotaon** command to turn on quotas for a file system. Example 5-46 shows an example of turning on quotas for an already mounted file system called `/vrtfs`. To turn on user quotas only, use the `-u` flag with the **vxquotaon** command. To turn on group quotas only, use the `-g` flag with the **vxquotaon** command. If you are only using user quotas, the `quotas.grp` file does not need to exist; also, if you are using only group quotas the `quotas` file does not need to exist.

*Example 5-46 Turning on quotas for an already mounted file system*

---

```
# cd /vrtfs
# touch quotas
# touch quotas.grp
# vxquotaon /vrtfs
```

---

Alternatively, quotas can be enabled at mount time. The `quotas` and `quotas.grp` file does not need to exist if enabled at mount time, as they will be created during the mount process. To enable both user and group quotas for a file system called `/vrtfs`, run the following command:

```
# mount -V vxfs -o quota /dev/vx/dsk/rootdg/mirrvol01 /vrtfs
```

To use the **mount** command to enable only user quotas for the same file system, run the following command:

```
# mount -V vxfs -o userquota /dev/vx/dsk/rootdg/mirrvol01 /vrtfs
```

To enable group quotas only with the **mount** command, run the following command:

```
# mount -V vxfs -o groupquota /dev/vx/dsk/rootdg/mirrvol01 /vrtfs
```

### Editing quotas

Quotas can be set and edited for users and groups by using the **vxedquota** command. Quotas do not need to be turned on for a file system in order to be able to edit them; however, they will not take effect until the quotas for a file



system are enabled. User and group quotas are allocated in 1 KB blocks for file system space and inodes for number of files allowed on the file system.

When editing quotas, the following values can be set:

- ▶ Users soft limit: The lower limit a user has for files and blocks. If a user exceeds this limit, the time limit determines how long the user has to get below this limit.
- ▶ Users hard limit: The absolute limit a user is restricted to for files and blocks.
- ▶ Group soft limit: Same as a user's soft limit, but applies to all users in the specified group.
- ▶ Group hard limit: Same as a user's hard limit, but applies to all users in the specified group.
- ▶ Time limit: This determines the amount of time that a soft limit can be exceeded before further user quotas are declined. This is global for the entire file system.

Example 5-47 shows an example of setting a user quota of 11 MB on a file system called /vrtfs for a user called user1 and a file quota limit of 250 files. The following steps are performed:

1. The **vxedquota -g** command is run with the user ID specified. This brings up a list with all VxFS file systems that have quotas turned on in an editing session similar to vi. The soft limit and hard limits are specified in 1 KB block sizes. The soft limit and hard limit for inodes or number of files is then specified.
2. The **vxedquota -t** is then run to specify the time limit for blocks and files. These can be specified as month, week, day, hour, min, or sec.

*Example 5-47 Setting user quotas on a VxFS file system*

---

```
# vxedquota -g user1
.....
fs /vrtfs blocks (soft = 10000, hard = 11000) inodes (soft = 200, hard = 250)
vxedquota -t
# vxedquota -t
.....
fs /vrtfs blocks time limit = 1.00 day, files time limit = 1.00 day
```

---

## Viewing quota information

There are two commands for viewing quota information for VxFS file systems: **vxquot** and **vxquota**. The **vxquot** command is used to view a particular user's or group's disk usage of a file system. Example 5-48 on page 178 shows the output from the **vxquot** command used to report on all users quotas of a file system. This shows user1 as having used 10358 1 KB blocks and 27 files.

#### Example 5-48 Displaying file system usage using vxquot

```
# vxquot -f /vrtfs
/dev/vx/rdisk/rootdg/mirrvol01:
USERS
35148      25   root
10353      27   user1
```

Example 5-49 shows the output of the **vxquota** command used to find out how much space and number of files user1 is using. Note that user1 has exceeded the soft limit for number of 1 KB blocks and has 59.7 minutes to reduce this below 10000 blocks before user1 is unable to allocate any more space on the file system.

#### Example 5-49 Displaying a user's quota information

```
# vxquota -v -u user1
Disk quotas for user1 (uid 201):
Filesystem      usage  quota  limit  timeleft  files  quota  limit  timeleft
/vrtfs           10763  10000  11000   59.7 mins    35    200   250
```

### Turning off quotas

Quotas can be turned off by using the **vxquotaoff** command. There are two flags with this command: -u, to turn off user quotas, and -g, to turn off group quotas. If a flag is not specified, both user and group quotas are turned off. If quotas are turned off, the current quota limits set by the **vxedquota** command are not enforced until quotas are turned on again.

To turn off quotas for a file system called /vrtfs, run the following command:

```
# vxquotaoff /vrtfs
```

## 5.6.3 Defragmenting file systems

Over time a file system may become fragmented, with the space allocated to files changing as they are removed, added, and changed. This may cause gaps in the extent allocation. The **fsadm** command can be used to rearrange the extent allocation and reduce fragmented space on a VxFS file system by making the space allocated to small files contiguous, and reducing the free space in the extents allocated to smaller files, which may in some cases increase the available space in the file system.

Example 5-50 on page 179 shows a report being run on fragmentation on a file system called /vrtfs. This example shows the fragmentation levels to be normal. To determine the fragmentation of the file system, the lines showing the percentage of free blocks should be viewed.

For optimal space allocation, the free file system space should be in the larger extents while the smaller extents should have very little free space. The following guidelines are used to determine whether a file system is fragmented:

- ▶ The Dirs to Reduce column has a large number.
- ▶ More than one percent of free space exists in extents smaller than 8 blocks in length
- ▶ More than five percent of free space exists in extents smaller than 64 blocks in length
- ▶ Less than five percent of the total file system available space is in extents larger than 64 blocks or more

*Example 5-50 Reporting on file system fragmentation*

---

```
# fsadm -D -E -s /vrtfs
```

Directory Fragmentation Report

|       | Dirs<br>Searched | Total<br>Blocks | Immed<br>Dirs | Immeds<br>to Add | Dirs to<br>Reduce | Blocks to<br>Reduce |
|-------|------------------|-----------------|---------------|------------------|-------------------|---------------------|
| total | 14               | 12              | 5             | 0                | 0                 | 0                   |

Extent Fragmentation Report

| Total<br>Files | Average<br>File Blks | Average<br># Extents | Total<br>Free Blks |
|----------------|----------------------|----------------------|--------------------|
| 239            | 229                  | 1                    | 46310              |

blocks used for indirects: 0

% Free blocks in extents smaller than 64 blks: 0.50

% Free blocks in extents smaller than 8 blks: 0.06

% blks allocated to extents 64 blks or larger: 95.24

Free Extents By Size

|             |   |             |   |            |   |
|-------------|---|-------------|---|------------|---|
| 1:          | 4 | 2:          | 3 | 4:         | 5 |
| 8:          | 3 | 16:         | 3 | 32:        | 4 |
| 64:         | 2 | 128:        | 1 | 256:       | 1 |
| 512:        | 1 | 1024:       | 2 | 2048:      | 1 |
| 4096:       | 0 | 8192:       | 1 | 16384:     | 0 |
| 32768:      | 1 | 65536:      | 0 | 131072:    | 0 |
| 262144:     | 0 | 524288:     | 0 | 1048576:   | 0 |
| 2097152:    | 0 | 4194304:    | 0 | 8388608:   | 0 |
| 16777216:   | 0 | 33554432:   | 0 | 67108864:  | 0 |
| 134217728:  | 0 | 268435456:  | 0 | 536870912: | 0 |
| 1073741824: | 0 | 2147483648: | 0 |            |   |

---

To defragment a file system, run the following command:

```
# fsadm -d -e -s /vrtfs
```

The -d option will perform directory defragmentation while the -e option will perform extent defragmentation. The -s option is used to print a summary.

File systems can be defragmented via the VEA by selecting the file system in the file system view window and then selecting **Actions -> Defrag File System**.

## 5.6.4 Optionally licensable features

The following list contains some licensable features of VxFS and a brief description of each:

- ▶ **Storage Checkpoints:** This enables a snapshot of a file system to be taken, but does not require an additional volume for the snapshot, and can be used as the basis of block level incremental backups.
- ▶ **Quick I/O for databases:** This is used for applications such as Oracle to enhance database performance.
- ▶ **QuickLog:** Enhances file system performance and recovery time.

For more information on these features, see the *VERITAS File System 3.4 - Administrators Guide (AIX)*.

## Comparisons

This chapter reviews VERITAS Foundation Suite™ for AIX, concentrating on AIX as the underlying operating system. There are two focus areas:

- ▶ The first considers specifics for VERITAS Foundation Suite on AIX and, where appropriate, differences compared with other platforms are highlighted. This includes coexistence and interoperability with common components that you might see in IBM @server pSeries solutions, such as subsystem device driver for the Enterprise Storage Server (ESS).
- ▶ The second is a review of the functional comparisons between the AIX LVM and JFS/JFS2 and VxVM and VxFS.

## 6.1 Comparisons with other UNIX platforms

Changes were made in the AIX kernel to support the port of VERITAS Foundation Suite to AIX and as a result of the port there are some differences in the Foundation Suite on AIX compared with other platforms. It is useful as a system administrator to understand where these differences exist.

### 6.1.1 ODM and SMIT integration

The Object Data Manager (ODM) is unique to AIX and is used to manage and manipulate system, device, and application information in an object oriented database. The ODM provides a standard way for handling the configuration and customization of system resources and devices. It is robust, secure, and easily shareable compared with traditional UNIX use of ASCII stanza and colon files. Predefined information is populated in the ODM classes when an AIX system is installed and as changes are made, specific to the system, the customized object classes are updated. The default directory for the ODM is held in the ODMDIR variable and is usually set to /etc/objrepos.

The installation of VERITAS Foundation Suite for AIX adds entries for each of the VERITAS device drivers (VxIO, VxSPEC, VxPortal, VxQIO, and VxDMP) to the predefined devices object class (PdDv). Then at reboot time or upon execution of the configuration manager, **cfgmgr**, the configuration method that is listed in the predefined stanza for that class or type, will be run. It is possible, although very rarely required, to view the contents of an ODM class. To view the entries in the PdDv object class, run the following command:

```
# odmget PdDv | grep -p vx | more
```

The PdDv ODM stanza for the VxIO Device Driver is shown in Example 6-1.

*Example 6-1 PdDv ODM stanza for the VxIO device driver*

---

```
#odmget PdDv | grep -p vxio
PdDv:
    type = "vxio"
    class = "vxdrv"
    subclass = "node"
    prefix = "vxio"
    devid = "0"
    base = 0
    has_vpd = 0
    detectable = 0
    chgstatus = 1
    bus_ext = 0
    fru = 0
    led = -17470
```

```

setno = 1
msgno = 1
catalog = "vxvm.cat"
DvDr = "vxio"
Define = "/etc/methods/defvxio"
Configure = "/etc/methods/cfgvxio"
Change = "/etc/methods/chgvxio"
Unconfigure = "/etc/methods/ucfgvxvm"
Undefine = "/etc/methods/undefine"
Start = ""
Stop = ""
inventory_only = 0
uniquetype = "vxdrv/node/vxio"

```

---

Support for basic VxVM commands is also provided through the system management interface tool (SMIT), in addition to the VxVM command line or menu based tools such as **vxdiskadm**. Installation of VxVM populates the ODM classes related to SMIT with VxVM menu support. In the `/etc/objrepos` ODM directory, the SMIT object classes begin with `sm_`.

The main SMIT VxVM window shown in Example 6-2 can be reached by typing the **smitty vxvm** fast path on the command line. VxVM tunables can be edited via SMIT, and also have stanzas in the ODM predefined attributes (PdAt) class. Changes to tunable values will take effect on reboot or reload of VxVM kernel extensions.

*Example 6-2 Main SMIT menu for VERITAS Volume Manager*

---

```

VERITAS Volume Manager

Move cursor to desired item and press Enter.

Disk Groups
VxVM Volumes
VxVM Disk Administrator
Change / Show VxVM Tunables
Configure Replicated Data Set (RDS)
Replication Tasks

F1=Help      F2=Refresh   F3=Cancel    F8=Image
F9=Shell     F10=Exit     Enter=Do

```

---

## 6.1.2 Disk devices and the VxVM

On AIX systems, physical disks are represented as an `hdisk#`, where `#` is unique integer assigned when the disk is first detected. Compare this with Sun Solaris:

```
[On Solaris system]
# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
  0. c0t0d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
    /pci@1f,4000/scsi@3/sd@0,0
  1. c0t1d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
    /pci@1f,4000/scsi@3/sd@1,0
```

Where `c#` is the controller, `t#` is the target ID, and `d#` is the disk number.

The controller name is a string on AIX, such as `scsi#`, `fscsi#`, or `ssar`, and can be identified using the `getctlr` option of the `vxdisk` command. An example would be:

```
[On AIX system]
# vxdisk getctlr hdisk4
scsi3
```

On a Sun Solaris system, `vxdisk getctlr disk_device` is not implemented in VERITAS Foundation Suite for Solaris, Version 3.4 and returns the following:

```
[On Sun Solaris system]
# vxdisk getctlr
vxvm:vxdisk: ERROR: Not implemented
```

**Note:** The `ssar` on AIX is the SSA adapter router and is a logical device. There is only one `ssar`, even if there is more than one SSA adapter. The adapters are named `ssa#`. SSA stands for Serial Storage Architecture.

### Enclosure-based naming

A further disk related difference for the Foundation Suite for AIX is that the VxVM uses enclosure-based naming by default. An example of enclosure based naming can be seen in 2.2.4, “Enclosure-based naming” on page 18.

### Supported VxVM disk types

Only simple and `nopriv` disk types are supported on AIX. Sliced disks are not supported by AIX, so are not considered by VxVM. The supported VxVM disk types are shown in Table 6-1 on page 185.



Table 6-1 Supported VxVM disk types Solaris and AIX

| Operating system/disk type | IBM AIX 5L | Sun Solaris |
|----------------------------|------------|-------------|
| simple                     | yes        | yes         |
| nopriv                     | yes        | yes         |
| sliced                     | no         | yes         |

VxVM disks with type nopriv do not have a private regions. nopriv disks are used for special purposes, such as a RAM disk. A private region is small area on the physical storage device where the disk label and other VxVM configuration information is held. The VxVM private region offset on a AIX disk starts at 256 block offset.

### vxresize

**vxresize** is a VxVM command that can be used to grow or shrink a VxVM volume and a file system. Resizing of a VxFS file system can only take place while it is mounted. Growth of JFS and JFS2 file systems are also supported on AIX. This is in line with the support for growth of a UNIX file system on a Sun Solaris system. Table 6-2 is a summary of supported operations.

Table 6-2 vxresize file system summary

| File system type/state | VxFS          | JFS  | JFS2 |
|------------------------|---------------|------|------|
| Mounted                | Grow / Shrink | Grow | Grow |
| Unmounted              | Not allowed   | Grow | Grow |

An unmounted JFS or JFS2 file system must have a valid entry in /etc/filesystems.

## 6.1.3 VxVM and LVM co-existence

IBM has provided co-existence Application Programming Interfaces (APIs) for VxVM and LVM that allow manipulation of the LVM record on disk. These are:

- ▶ set\_disk\_vm\_owner()
- ▶ qry\_disk\_vm\_owner()
- ▶ clr\_disk\_vm\_owner()

These interfaces are provided in both API and CLI format. The LVM record (stored in sector 7) must be cleared before the set\_disk\_vm\_owner interface is called to initialize the disk for use by VxVM. The **chpv -C hdisk#** command will

clear the LVM record, as described in 4.3.1, “Adding disks” on page 89. Upon initialization the VxVM version is stamped in sector 7 on the physical disk. When a disk is removed from VxVM control, the `clr_disk_vm_owner()` interface is called to clear the VxVM version from sector 7. This agreed interaction ensures that VxVM will not initialize physical volumes that are owned by the LVM.

Important disk sectors on AIX for the VxVM include:

|                  |  |
|------------------|--|
| <b>Sector 0</b>  | Contains the bootrecord and also the physical volume identifier. A description of what is in the bootrecord can be found in <code>/usr/include/sys/bootrecord.h</code> .   |
| <b>Sector 7</b>  | A region (sector) on disk that is used by the LVM when a physical volume is part of a volume group. It contains LVM ID field, size of the volume group descriptor area (VGDA), and the location that is reserved for bad blocks. See <code>/usr/include/lvmrec.h</code> for details. |
| <b>Sector 70</b> | A copy of the LVM record.  |

For detailed information about other reserved sectors on a physical disk attached to AIX, see the redbook *AIX Logical Volume Manager A-Z: Introduction and Concepts*, SG24-5432.

## 6.1.4 VxVM at system startup and shutdown

At system startup and after the system is up and running, the `/etc/inittab` file on AIX supplies script input for the `init` command, which handles the dispatching of processes. Under AIX, entries in `/etc/inittab` have the following format:

Identifier:RunLevel:Action:Command

VxVM rc scripts are placed in the `/etc/init.d` directory and entries are added to the `/etc/inittab` during installation of the VxVM. An extract from `/etc/inittab` is shown in 4.1.1, “Startup process” on page 79. No VxVM related entries are placed in `inittab` on a Sun Solaris system.

VERITAS Foundation Suite for Solaris has two shutdown scripts:

- ▶ `/etc/rc0.d/K99vxvm-shutdown`
- ▶ `/etc/rc0.d/K10.vmsa-server`

These do not exist on an AIX platform.

Some scripts are installed on AIX that are not required. For example, `/etc/rcS.d/S35vxvm-startup1`, which is for startup of a startup of an encapsulated root file system, is not run under AIX, although a copy of the script can be found

on AIX. This reference is included for completeness; it does not affect the running of VERITAS Foundation Suite for AIX.

### 6.1.5 VxVM/VxFS command differences

There are a relatively small number of differences at the command level on an AIX platform compared with Sun Solaris. These include:

- ▶ The use of the **vxlicinst** and **vxlicrep** commands for licensing on AIX and **vxlicsense** on Sun Solaris.
- ▶ The option in **mkfs** to provide a file system type is **-F vxfs** on Solaris and **-V vxfs** on AIX. This is determined by the operating system, not VxFS, but it is useful for administrators of these two environments to be aware of the difference.

This is not an exhaustive list, but highlights differences for some frequently used commands.

### 6.1.6 VxVM/VxFS device drivers and kernel extensions

As we have seen, installation of the VERITAS Foundation Suite results in entries being added to the ODM. The class for the entries added is *vxdrv*. The AIX list devices command **lsdev** can then be used to see the state of the VERITAS device drivers:

```
# lsdev -Cc vxdrv
vxdump    Available  VERITAS VxDMP Device Driver (3.2p)
vxio      Available  VERITAS VxIO Device Driver (3.2p)
vxspec    Available  VERITAS VxSPEC Device Driver (3.2p)
vxportal0 Available  VERITAS VxPORTAL Device Driver
vxqio0    Available  VERITAS VxQIO Device Driver
```

The VxVM and VxFS device driver (kernel extensions) are located in the */usr/lib/drivers* directory:

```
# ls -la /usr/lib/drivers | grep -i vx
-r-xr-xr-x  1 root    sys      471677 Apr 26 22:01 vxdmp
-r-xr-xr-x  1 root    system    4183357 Apr 22 12:55 vxfs.ext
-r-xr-xr-x  1 root    sys      4053270 Apr 26 22:01 vxio
-r-xr-xr-x  1 root    system    32547 Apr 22 12:55 vxportal.ext
-r-xr-xr-x  1 root    system    242826 Apr 22 12:55 vxqio.ext
-r-xr-xr-x  1 root    sys      30285 Apr 26 22:01 vxspec
```

The **vxkextadm** command can be used to determine the status of VxFS kernel extensions; these include vxqio, vxfs, vxportal, and ted. To check the status of the qio kernel extension, run:

```
# ./vxkextadm qio status
```

Kernel extension /usr/lib/drivers/vxqio.ext is not loaded

or to load/unload the kernel extension:

Usage: vxkextadm qio load|unload|status

Alternatively, the **genkex** command, which is part of the bos.perf.tools fileset, can be used to view all currently loaded kernel extensions:

```
# genkex | grep vx
2047380          149c /usr/lib/drivers/vxspec
5ffb000          150200 /usr/lib/drivers/vxio
2028840          1eb38 /usr/lib/drivers/vxdmp
2008f80          ad78 /usr/lib/drivers/vxqio.ext
2008098          ed0 /usr/lib/drivers/vxportal.ext
5e0b000          113214 /usr/lib/drivers/vxfs.ext
```

All the methods to configure/unconfigure VxVM kernel extensions are located in /usr/lib/methods, which is also linked from /etc/methods. The **ucfgvxvm** method can be used for unconfiguring the VxVM modules on AIX. Under AIX, the **rmdev** command will invoke the required **ucfgvxvm** method and typically provides more informative feedback should the command fail.

An example of using both **rmdev** and the **ucfgvxvm** method explicitly to unconfigure a VERITAS device driver is shown next. Take the VxVM device driver, vxspec, as an example. First, check the current status of vxspec:

```
# lsdev -Cc vxdrv | grep vxspec
vxspec    Available  VERITAS VxSPEC Device Driver (3.2p)
```

Run the **ucfgvxvm** method explicitly. The command appears to have completed without any problems, as there is no further output; however, when **lsdev** is run again, the vxspec device driver is still in the available state:

```
# /usr/lib/methods/ucfgvxvm -l vxspec
# lsdev -Cc vxdrv | grep vxspec
vxspec    Available  VERITAS VxSPEC Device Driver (3.2p)
```

If the AIX command **rmdev** is called, we get a method error output, which tells us that this device driver is busy:

```
# rmdev -l vxspec
Method error (/etc/methods/ucfgvxvm):
0514-062 Cannot perform the requested function because the
specified device is busy.
```

If the **ucfgvxvm** method is run on the VxFS device driver vxqio, then this will work:

```
# lsdev -Cc vxdrv | grep vxqio
vxqio0    Available  VERITAS VxQIO Device Driver
```

Run the **ucfgvxvm** method:

```
# /usr/lib/methods/ucfgvxvm -l vxqio0
# lsdev -Cc vxdrv | grep vxqio
vxqio0    Defined    VERITAS VxQIO Device Driver
```

Make the vxqio0 kernel extension available once more with either the **mkdev -l** command or the **cfgvxvm** method:

```
# mkdev -l vxqio0
vxqio0 Available
```

This time, use **rmdev -l** to bring the device into the defined state:

```
# rmdev -l vxqio0
vxqio0 Defined
```

This time, there is a confirmation of the action to bring the device into the defined state.

It is clear that the VERITAS device drivers must be stopped and started in a particular order and with the relevant VERITAS daemons in an appropriate state. This procedure is described in 4.1, “System startup and process control” on page 79.

There should be at least one vxiod kernel thread in the system all the time. Should this not be the case, run:

```
# vxiod
10 volume I/O daemons running
# vxiod set 10
```

## 6.1.7 Installation and packaging

Installation and packaging of VERITAS Foundation Suite for AIX has been discussed in Chapter 1, “Introduction” on page 1 and Chapter 3, “Planning and installation” on page 41. Installation images are provided in **installp** format on an AIX platform. Installation can be undertaken via SMIT, from the command line with **installp**, or using the VRTSinstall script. All of the standard AIX commands to list information about licensed product packages (LPPs) work for VRTS packages, including:

- ▶ **ls1pp -L VRTS\***: Lists all installed VRTS packages. This command should not be run from a directory that contains install images.
- ▶ **1ppchk -v [-c]**: Licensed product package check tool. Both the -v and -c options should return cleanly with no output. Any differences should be noted and confirmed that they are as expected, for example, **1ppchk -c** may return a difference in size for a script that an administrator has edited.

- ▶ **lslpp -w /usr/bin/vxdg**: Returns the name of the fileset that contains the command.
- ▶ **lslpp -f fileset\_name**: Displays the names of the files added to the system during installation of the specified fileset.

## 6.1.8 The 64-bit kernel

The kernel on an AIX server is /unix, which is linked to a file in the /usr/lib/boot/unix\_\* file.

On AIX, there is a single 64-bit kernel for use on either uniprocessor or multiprocessor servers. On a 64-bit system, /unix is linked to /usr/lib/boot/unix\_64; the **bootinfo -K** command will show whether the running kernel is 32-bit or 64-bit.

VERITAS Foundation Suite for AIX will run on a 32-bit or 64-bit AIX kernel. There are 32-bit and 64-bit specific VxVM kernel extensions. VxVM utilities are all 32-bit; however, 64-bit applications can make calls to 32-bit VxVM drivers.

## 6.1.9 Debugging information

As part of the VxVM port to AIX, IBM has provided the following trace hooks for use with the AIX kernel **trace** command. These are:

- ▶ VxIO 620
- ▶ VxSPEC 612
- ▶ VxDMP 622
- ▶ VxFS 0E1

There is a single trace hook for VxFS.

There is no group name for the trace hooks associated with the VxVM and VxFS. To list all the trace hooks available on a system, you can run:

```
# trcrpt -j
```

The following commands could be used to collect VxVM trace data from the command line:

```
# trace -a -c -j 620,621,622
# execute task to be traced, for example, vxplex att testvol testvol-01
# trcstop
# trcrpt -d 620,621,622
```

The default location for trace output is /var/adm/ras/trcfile. The -c option in the preceding **trace** command saves any existing trace file before starting tracing.

Only events related to the trace hooks after the -j will be captured. An extract from the output of running the command above is included in Example 6-3.

*Example 6-3 Sample trace output for VxVM events*

```
[On AIX system]
# trcrpt -d 620,621,622 | more
Wed Aug 14 10:04:44 2002
System: AIX srvr80e Node: 5
Machine: 0001615F4C00
Internet Address:
The system contains 4 cpus, of which 4 were traced.
Buffering: Kernel Heap
This is from a 32-bit kernel.
Tracing only these hooks, 620,621,622

trace -a -c -j 620,621,622
```

| ID  | ELAPSED_SEC  | DELTA_MSEC | APPL          | SYSCALL   | KERNEL   | INTERRUPT |
|-----|--------------|------------|---------------|---|----------|-----------|
| 001 | 0.000000000  | 0.000000   |               |   | TRACE ON | channel 0 |
| ... |              |            |               |   |          |           |
| 620 | 71.119424547 | 0.076737   | VxIO driver   | pstart: rhdisk5   |          |           |
|     |              |            |               | pblock=11F8 (lbp,pbp)=(32176D00,32176E80) B_READ B_BUSY opts: |          |           |
| 622 | 71.122037780 | 2.613233   | VxDMP driver  | pend: pbp=31DB  |          |           |
|     |              |            |               | 4200 resid=0000 error=0000 B_WRITE B_BUSY                     |          |           |
| 622 | 71.122849692 | 0.811912   | VxDMP driver  | pend: pbp=31FB  |          |           |
|     |              |            |               | 6A80 resid=0000 error=0000 B_WRITE B_BUSY                     |          |           |
| 621 | 71.122881945 | 0.032253   | VxSPEC driver | exit ioctl: dev=2D0000,                                       |          |           |
|     |              |            |               | cmd=564F4CA1, arg=20385EE0, ret=0000, rval_p=F0017EA0         |          |           |
| 621 | 71.125097260 | 2.215315   | VxSPEC driver | enter ioctl: dev=2D0000,                                      |          |           |
|     |              |            |               | cmd=564F4C7F, arg=2FF22070, mode=0003, ext=0000               |          |           |

On AIX 5L, the Kernel Debugger (KDB) is provided for analysis of a system dump or the running kernel. Prior to AIX 5L, this facility was provided by a utility called crash. The kernel debugger will typically only be used at the request of technical support. Sub commands for use with kdb are placed in /usr/lib/ras/autoload at installation of the VRTSvxvm package. These are:

```
# ls -la /usr/lib/ras/autoload | grep vx
-rwx----- 1 root sys 29737 Apr 26 22:00 vxdrv32.kdb
-rwx----- 1 root sys 34061 Apr 26 22:00 vxdrv64.kdb
```

Sub commands can be registered by calling the name of the library from within kdb. The KDB Kernel Debugger must be loaded at boot time. See <http://www.ibm.com/servers/eserver/pseries/library/> and click on the link for AIX Version 5L documentation library for documentation on using the KDB Kernel Debugger and the AIX trace facility.

### 6.1.10 Dynamic MultiPathing (VxDMP)

Device related porting changes for AIX require that device specific information be retrieved from the ODM. VERITAS Volume Manager (VxVM) Version 3.2 introduced the device discovery layer (DDL) to make support and attachment of new disk arrays possible without a reboot. The DDL provides a device discovery service for the `vxconfigd` configuration daemon. The DDL discovers multipathing attributes of disks and disk arrays that are connected to the host system and also any enclosure information. On AIX, the DDL gets device specific information from the ODM and, after discovery, downloads a device map into the VxDMP kernel. A different dynamically loaded library is used by the DDL to gather device attributes for each different disk array type. These libraries are known as Array Support Libraries (ASLs).

DDL uses SCSI commands to discover disk array attributes. A new SCSI pass-through interface was provided by IBM for VxVM. This SCSI pass-through is only available on AIX 5L Version 5.1 maintenance level 01 or later. The DDL layer can be administered with the `vxddladm` command. `vxddladm` has options to manage support for a particular type of disk subsystem and can also be used to exclude disks arrays from VxVM. To see the currently supported disk arrays on AIX, run the `vxddladm listsupport` command shown in Example 6-4.

*Example 6-4 Supported disk arrays*

|                        |            |         |             |
|------------------------|------------|---------|-------------|
| [On AIX system]        |            |         |             |
| # vxddladm listsupport |            |         |             |
| LIB_NAME               | ARRAY_TYPE | VID     | PID         |
| =====                  | =====      | =====   | =====       |
| libvxemc.so            | A/A        | EMC     | SYMMETRIX   |
| libvxhds.so            | A/A        | HITACHI | OPEN-*      |
| libvxhitachi.so        | A/PG       | HITACHI | DF400       |
| libvxhitachi.so        | A/PG       | HITACHI | DF500       |
| libvxpurple.so         | A/P        | SUN     | T300        |
| libvxshark.so          | A/A        | IBM     | 2105        |
| libvxvpath.so          | A/A        | IBM     | VPATH_NODES |

The IBM 7133 SSA array is supported by default. It is recommended that the appropriate ASL support be installed for a disk array if it is not listed in the `vxddladm listsupport` output. In some cases, only limited disk array function may be available without the ASL. ASL packages may be provided by third party array vendors and may also be available on VERITAS Software CDs.

In addition to this, ASLs are available for download from the VERITAS Technical Support site at <http://support.veritas.com>. Click on the **Knowledge Base Search** link, and select **Volume Manager for UNIX** from the **Product** drop-down menu. Enter the search phrase "VERITAS Enabled Arrays" (quotes must be



included in the search). Any newly issued ASLs can later be found at the same site.

The array support library development package is available on AIX for any future array library additions by array vendors.

### **VxDMP and Subsystem Device Driver (SDD) co-existence**

Multipathing on the IBM Enterprise Storage Server (ESS) is currently done by the Subsystem Device Driver (SDD). If you can see the same Logical Unit Number (LUN) in an ESS from multiple adapters, then you will get an `hdisk` for each path to the LUN. SDD creates a single `vpath#` device for each set of paths to disk and moves the physical volume identifier (PVID) from the `hdisk`s to the `vpath#` device. Volume groups are then comprised of `vpaths` rather than `hdisk`s. SDD load balances over the multiple physical paths to disk using the `vpath#` device as the entry point.

IBM has provided the following VxDMP co-existence APIs as part of the Sub System Device Driver on AIX 5L Version 5.1:

- ▶ `DMP_XCHG_GET_PATHS`
- ▶ `DMP_XCHG_GET_META`
- ▶ `DMP_XCHG_GET_ENCLR_INFO`

VxDMP and SDD controlled LUNs can co-exist. By default, the installation of the `VRTSvxvm` fileset installs the third party `vpath` library `/etc/vx/lib/discovery.d/libvxvpath.so`. The VxVM diagnostic utility `/usr/lib/vxvm/diag.d/vxdmpt` can be used for testing third party driver ioctls.

## **6.2 AIX LVM, JFS/JFS2 and VxVM, VxFS compared**

In 2000, the following whitepapers were published comparing LVM and VxVM, and JFS and VxFS. Publication of these documents were prior to the release of AIX 5L and the VERITAS Foundation Suite for AIX, Version 3.4, so this section of the redbook expands and updates this existing work.

- ▶ For *Quick Reference AIX Logical Volume Manager and VERITAS Volume Manager*, visit:  
[http://www.ibm.com/servers/aix/products/aixos/whitepapers/lvm\\_ver.html](http://www.ibm.com/servers/aix/products/aixos/whitepapers/lvm_ver.html)
- ▶ For *Quick Reference AIX Journaled Filesystems and VERITAS File System*, visit:  
[http://www.ibm.com/servers/aix/products/aixos/whitepapers/jfs\\_vfs.html](http://www.ibm.com/servers/aix/products/aixos/whitepapers/jfs_vfs.html)

Updated versions of the command comparison tables contained in the whitepapers are included in Appendix A, “LVM and VxVM command comparison tables” on page 299. The *VERITAS Volume Manager 3.2 - Migration Guide (AIX)*, which you can find at <http://seer.support.veritas.com/docs/246777.htm>, also has some of the contents of the whitepaper encapsulated in it.

The AIX LVM and VxVM discussion in this redbook approaches comparisons based on the following areas:

- ▶ Logical volume concepts
- ▶ Volume layouts
- ▶ Backup
- ▶ Hot spare management
- ▶ Concurrent access

Following this discussion, there is a comparison of JFS/JFS2 with VxFS.

## 6.2.1 Logical volume concepts

The AIX logical volume manager and VxVM both provide a powerful but usable interface to support the management of fixed storage. Each is comprised of logical and physical components that support a hierarchy of structures used to manage storage. The volume managers themselves control disk resources by mapping between the logical view of storage that is presented to the system administrator and the data physically on disk. Both volume managers are a collection of operating system commands, library subroutines, and other tools. Volume management with VxVM and LVM is described through physical and logical components.

### Physical disks

An LVM physical volume (PV) and VxVM disk (VM disk) are both representative of an underlying physical disk that is available for use by the respective volume managers. In AIX, when the system detects the disk device, it is initialized as a physical volume and assigned an `hdisk#`, number, where `#` is a unique integer. The device path is `/dev/hdisk#` for the block device or `/dev/rhdisk#` for the character device. A physical disk placed under the control of the VxVM is known as a VM disk and has a symbolic name associated with it for administrative purposes. The device path for a VM disk is `/dev/vx/dmp/da_name` for the block device or `/dev/vx/rdmp/da_name` for the character device. On an AIX platform the disk access (`da_name`) for the VxVM disk will be `hdisk#`, and `c#t#d#s#` on Solaris.

### ***Physical volume layout***

Both the LVM and VxVM partition the physical storage to ensure all the disk space can be utilized and to permit logical volumes and therefore file systems to span physical volumes.

In LVM, the following terms are used when talking about the layout of direct access storage devices or disks:

- ▶ **Block:** A 512 byte region that corresponds to a DASD sector. Block and sector are often used interchangeably.
- ▶ **Physical partition:** A physical partition (PP) is a contiguous set of blocks of fixed length within a single physical volume. The number of blocks in a partition and the number of partitions in a volume group are fixed when a volume group is created (see “Volume/disk groups” on page 195).

In VxVM the relevant terms are:

- ▶ **Block:** A 512 byte region that corresponds to a DASD sector. Block and sector are often used interchangeably. This is the same as with LVM.
- ▶ **Subdisk:** A contiguous set of blocks of arbitrary length within a single physical volume.

So in terms of physical component comparisons, we have the following:

- ▶ LVM physical volume (PV) is equal VxVM disk (VM disk).
- ▶ `/dev/[r]hdisk#` is equal `/dev/vx/[r]dmp/da_name`.
- ▶ Disk block is equal Disk block (same for both LVM and VxVM).

### **Volume/disk groups**

LVM volume groups and VxVM disk groups are collections of physical disks. This grouping allows multiple disks to be manipulated as a single piece of storage. Each volume group or disk group has a unique soft serial ID called the volume Group Identifier (VGID) in LVM and disk group ID (DGID) in VxVM. In LVM, the VGID is used by low level LVM commands. The DGID is used in the control and management of disk groups. The format of the VGID and DGID differ. The VGID is purely alphanumeric whereas the DGID encapsulates the host name of the system it was created on. For example:

```
# lsvg vg_name
...VG IDENTIFIER: 0001615f00004c00000000ef23ff961d
# vxdg list
...ID 1029264193.1170.srvr80e
```

Both volume managers use the physical disk to hold configuration information. There is no direct comparison between the reserved areas for the two volume managers. Definitions are included here to aid later discussions.

In LVM, there are three reserved areas on disk:

- ▶ **Volume Group Reserved Area:** The VGRA is comprised of the first 128 sectors on the disk are reserved. Details of what is held in these reserved sectors can be found in the header file `/usr/include/sys/hd_psn.h`. It is not expected that the system administrator will have any reason to access the VGRA.
- ▶ **Volume Group Status Area:** The VGSA exists when a physical volume is part of a volume group. The VGSA is used to store information about partitions when the data held in the partition is out of date. The VGSA is also used for quorum.
- ▶ **Volume Group Descriptor Area:** The VGDA holds information physically on disk about the volume group, physical volumes, and logical volumes. The VGDA is also used for quorum voting.

In VxVM, there is only one reserved area by default; as we will see later, other areas on disk can be allocated for specific purposes:

- ▶ **Private region:** The private region contains VxVM metadata, such as the disk label and other configuration records. It can vary in length, but is at offset 256 on AIX.
- ▶ **Configuration copy:** An area within the private region. Each disk group has a number of configuration copies written to a subset of the disks which comprise the disk group. Configuration copies contain disk group information. This information is used to populate the configuration database in the `/etc/vx/tempdb` directory. Note this directory is in `/var/vxvm/` on Solaris.

### ***Disk independence***

Volume management under both LVM and VxVM is independent of disk type, so multiple different disks with potentially different sizes can be grouped together.

### ***Maximum number of physical volumes***

The maximum number of physical volumes supported by AIX is 128 for “big” volume groups. The default maximum for other volume groups is 32 physical volumes. There is no maximum number of physical volumes for a VxVM disk group.

### ***Volume/disk group configuration information***

When LVM volume groups are created or imported, information from the volume group descriptor area (VGDA) is stored in the Object Data Manager (ODM) database. In addition, some files in `/etc` directory contain information that relates to volumes groups, for example, `/etc/vg`. AIX commands such as `lspv` and `lsvg`, use both the ODM and the VGDA on disk. There is always at least one copy of the VGDA on each disk in a volume group. The ODM is unique to LVM.

The VxVM retrieves disk group information from the physical disk only and populates a configuration database that is held on the AIX server in `/etc/vx/tempdb`. The area on the physical disk is called the configuration copy. By default, a configuration copy is allocated for each controller in a disk group. The number of configuration copies can be set with the `nconfig=number_of_copies` option of the **`vxvg init`** command. The same option can be used to change the number of configuration copies in an existing disk group with the **`vxdit`** command. All VxVM configuration changes have a transaction ID, sequence number, and timestamp associated with them. So during an import, if there is a difference in configuration copies, the disk with the latest copy will be used as the basis for the import. If there are not enough valid configuration copies (in the same way that AIX will not bring a volume group on line that has insufficient good VGDA's), VxVM will fail the import.

### ***Managing access to physical devices***

Volume manager commands control host access to the physical disks to prevent divergence or corruption of data that might occur if multiple systems have access to the disks at the same time. This is typically handled at the volume/disk group level.

There is the concept in LVM of bringing a volume group online, at which time reserves will be set on the disks that comprise the volume group. The **`varyonvg`** command is used to bring a volume group online and **`varyoffvg`** to take it offline, unsetting the reserves on the disks. An LVM volume group can be known about by multiple systems, although typically it is only accessed by one system at a time. Any other systems that can see the disks will be unable, by default, to access the disks or change them. It is possible to temporarily unset the locks to allow another server to read the volume group configuration information. In LVM, an import of a volume group automatically brings the volume group online unless otherwise directed.

Similarly in VxVM, locks are written to disk when a disk group is imported with the **`vxvg import`** command or at disk group creation. A disk group cannot be imported onto two systems at the same time. Only when the disk group is deported are the reserves unset on the disks. There is no equivalent concept to varying on/off a volume group. At each import, the configuration copies must be read from disk. This would be required following a system reboot. If a disk group is moved to another system following a system crash, then the **`vxvg -C import dg_name`** command can be used to force the release of the reserves that have been left set on the disks when the system failed. There is no concept of temporarily unsetting reserves.

## AIX logical volumes and VxVM volumes

Logical volumes (AIX logical volume or VxVM volume) present what appears to be contiguous space on disk to the application layer, but can be non contiguous on the physical disk.

In LVM, the following terms are used when talking about the layout of logical volumes:

- ▶ Logical partition: A logical partition (LP) maps to a physical partition, or for mirroring, up to three further physical partitions.
- ▶ Logical volume: A set of logical partitions that represents a piece of space storage.

In VxVM the relevant terms are:

- ▶ Plex: A set of subdisks, possibly from multiple physical volumes.
- ▶ Volume: A set of plexes that represent a piece of storage

For both LVM and VxVM, logical volumes can only exist within a single volume group or disk group. They cannot be mirrored or expanded into other volume or disk groups.

### ***Types of logical volume/VxVM volume***

Logical volumes can be unstructured (raw logical volumes) where the application or operating system will handle access to the data or used with a file system that provides structured access to data. Non typical logical volume types in LVM include boot, dump, copy, and jfslog. In VxVM, there are data change object (DCO) log volumes and also, at a lower level, the log subdisk.

A summary of physical and logical component comparisons for LVM and VxVM would be:

- ▶ Physical partition is equal subdisk
- ▶ Volume group is equal Disk group
- ▶ Logical partition is equal Plex
- ▶ Logical volume is equal VxVM volume

A pictorial summary of LVM components aligned with VxVM components is included in Figure 6-1 on page 199 to show, at its simplest, how a logical volume is built in the respective environments.

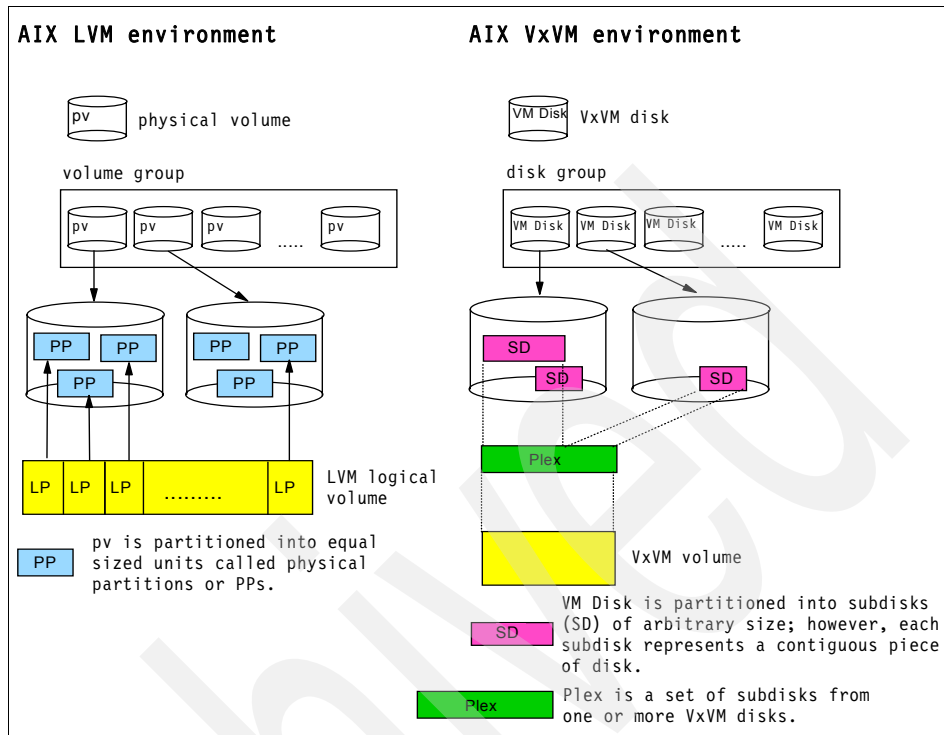


Figure 6-1 LVM and VxVM logical and physical components

Chapter 2, “Components” on page 9 reviews the VxVM components and includes examples of how more complex VxVM objects can be constructed.

## Device discovery

Device discovery without the need to reboot is supported by both the AIX LVM and VxVM. The `cfgmgr` executable on AIX can be run globally or from a specified controller to detect and configure the devices attached to that controller. Equivalent functionality is provided in VxVM via the device discovery layer (DDL).

## Software RAID levels

Different volume layouts are required to support management of storage and are typically described by a RAID level. RAID stands for Redundant Array of Independent Disks. There are five common levels that describe functions in terms of availability and performance. Table 6-3 on page 200 summarizes the supported software RAID levels for both LVM and VxVM.

Table 6-3 Supported software RAID levels for LVM/VxVM

| RAID level                      | LVM | VxVM |
|---------------------------------|-----|------|
| RAID 0 (striping)               | yes | yes  |
| RAID 1 (mirroring)              | yes | yes  |
| RAID 1+0 (mirroring + striping) | yes | yes  |
| RAID 5 (striping + parity)      | no  | yes  |

IBM @server pSeries servers support special adapters that implement RAID in hardware, which allows for the parity calculations to be handled more efficiently. Even though VxVM offers the RAID 5 functionality in software, it is not supported for shared storage, which is typically used in a high availability environment. Space is required on the VxVM disks in a RAID 5 configuration to support the RAID function. There are also special requirements with respect to setup of logs for the RAID 5 software configuration.

## 6.2.2 Volume layouts

This section describes and compares the LVM/VxVM features and options that are used to create volume layouts to support the software RAID levels described in Table 6-3.

### Mirroring

LVM and VxVM support multiple copies of a logical volumes. In LVM, three copies or mirrors are supported for every logical partition. In VxVM, there can be up to 32 copies of the volume. In this case, there would be multiple mirrored plexes. Where the application is read intensive, there are possible performance gains to be made from having greater than three mirror copies; however, experience shows that system administrators typically implement two or three copies. Three mirror copies is considered good practice, allowing for one copy to be split off for backup purposes, leaving two for availability.

The term logical volume and volume will be used interchangeably in this section. Note, however, that only logical volumes can be mirrored. Both LVM and VxVM have default options for the creation of a volume that will work for the majority of cases; however, it is still useful to understand the features and options for a logical volume that can affect performance, data integrity, and availability. The first consideration is the allocation policy, that is, where on the disk each copy will reside.



## Allocation policy

The allocation policy is important primarily to ensure that mirror copies reside on different physical disks for redundancy. The system administrator does not need to be concerned with how VxVM/LVM practically carries out this task, but should understand the result of the basic options in this area.

In LVM, the following allocation policy types apply:

- ▶ **strict**: This is the *default* allocation policy for LVM. All partitions with the same data will be placed on a separate physical volumes.
- ▶ **notstrict**: Copies from a logical volume can share the same physical volume. If the request for strict allocation cannot be met, then it would be appropriate to reorganize the disks or to add additional disks to enable the strict allocation to be met.
- ▶ **super strict**: For use with mirroring and striping. It was introduced in AIX Version 4.3.3. See “Striping” on page 205 for more details.

In VxVM:

- ▶ **ordered**: An ordered allocation policy is used by default from VERITAS Volume Manager (VxVM) 3.2. Ordered allocation can be applied explicitly, enabling the system administrator to control how mirrors and columns are laid out when creating a volume.

If the `-o ordered` option is specified when using `vxassist` to create a volume, any storage that is specified is allocated in the following order:

- a. Concatenate disks
- b. Form columns
- c. Form mirrors

At a more granular level, it is possible to manipulate placement of data on a disk with the *inter* and *intra* policies in LVM and *offset* option for subdisks in VxVM, but these are not considered here.

## Mirror read/write policy

In a mirrored environment, there are multiple copies of data, so the question arises, which copy should be read/written to? What are the effects of the mirror read/write policy on performance and data integrity?

In LVM, there are two types of mirror *read/write* policy:

- ▶ **Parallel**: The default for AIX is parallel, and in this case, writes are issued at the same time. Control is returned to the program when the write to the last disk has completed. Reads will use the copy from the disk with the least outstanding I/Os. This improves performance of mirrored writes. There is a

small chance that data integrity will be compromised if there is a failure during the write. Mirror write consistency can be set on to prevent any issues with data integrity. Mirror write consistency checking is discussed in a following section.

- Sequential: With a sequential mirror policy, there is the concept of a primary and secondary copy. Writes are made to the primary copy and then the secondary copy. Control is returned to the program only after all copies have been updated in turn. Reads will always be made from the primary copy in the first instance. Sequential writes ensure data integrity with a small compromise on performance.

In VxVM, there are no mirroring options; at volume creation, that determines mirror read/write policy. There are, however, implicit policies in place, and the read policy can be modified after creation:

- Parallel write policy: The default and only mirror write policy available in VxVM. Writes to copies are executed in parallel and control is returned to the application once all the writes have completed.
- Mirror read policy: The mirror read policy can be set explicitly for a volume using `vxvol rdpol`.
  - round: The round algorithm uses a list of mirrors and associated disks and traverses the list each time a non-sequential I/O request is made using an alternative copy for each read request. Sequential I/O requests are serviced from a single plex (copy).
  - prefer: Results in reads from a preferred mirror copy (plex). If this request cannot be met from the preferred copy, then the round algorithm will be used.
  - select: The *default* read policy will choose either round or prefer, depending on the volume configuration.

### **Failure of a mirror copy of data**

Occasionally, a mirror copy of data may become unavailable due to, for example, a temporary disk access problem, such as a loose disk cable or other environmental issue. During any temporary outage, a copy may become out of date or stale. Redirection of I/O is handled automatically should there be a failure to a mirror copy in both LVM and VxVM.

### ***Resynchronization and mirror write consistency***

In LVM, the device driver recognizes that there was a failure in fulfilling a write request and marks the partition as stale in the VGSA. The LVM device driver will stop sending any I/O requests to partitions marked as stale. When the problem has been resolved, the information in the VGSA is used to synchronize only those partitions marked as stale. This is the default behavior of LVM.

In VxVM, a failed write will be serviced from an available copy and the failure reported by `vxnotify`. The `vxinfo` command can be used to determine the status of a volume's plexes.

It is essential that the behavior of a mirrored environment is understood should there be a system failure during the write of a mirrored copy. This is of greater concern where the mirror writes have been scheduled in parallel. If there is a crash during the mirror write process, then this raises the question, are the mirror copies the same? Subsequent reads from different copies might return different values unless some form of mirror write consistency is employed. Volumes that have no I/Os pending at the time of the crash will not require any synchronization.

In LVM, the following policies affect resynchronization and mirror write consistency:

- ▶ Default LVM recovery behavior with parallel mirror write policy: If there is a system crash during a mirrored write, then it is not known which mirror has the latest copy of the data. The first mirror copy found will be used to synchronize the other copies.
- ▶ Mirror write consistency checking (MWCC): An option of an LVM logical volume and aids identification of logical partitions that may be inconsistent following a system crash. With MWCC, enabled write requests for the logical volume are held in the scheduling layer until the mirror write cache (MWC) blocks on disk can be updated. Only when the MWC blocks have been written does the write proceed. When the volume group is varied back online following an unexpected system outage, the information in the MWC blocks on each of the disks is used to make the logical partitions consistent again.
  - active: This is the *default* option for creation of a mirrored logical volume and will typically be used for mirrored logical volumes.
  - passive: This is a new option with AIX 5L Version 5.1.
  - off: MWCC is sometimes turned off to leverage additional performance or if the application requires that it is turned off.

MWCC does not guarantee that all writes will be completed, but it does mean that all the mirrors with a parallel scheduling policy will be the same following a system crash. For a detailed example of MWCC, see redbook *AIX Logical Volume Manager A-Z: Introduction and Concepts*, SG24-5432.

- ▶ Default LVM recovery behavior with sequential mirror write policy: As there is a known primary copy, this will always be read first following an unexpected system outage. This first copy will be used as the source to resynchronize other mirror copies, so there is an increased chance that all the mirrors have the latest copy of the data.

In VxVM, when a write is first issued to a volume, the whole volume is marked as *dirty*. The volume state is returned to *clean* when all writes have completed or the volume is closed. So after a system failure, only those volumes that appear to be dirty are resynchronized.

The following features/policies affect resynchronization and mirror write consistency:

- Default VxVM recovery behavior with parallel mirror write policy: This is known as *read-writeback* recovery mode. If there is a system crash during a mirrored write and the volume is marked as dirty, it is not known which mirror has the latest copy of the data. One of the mirror copies must be chosen to synchronize the others. Synchronization of data is done in the background. Reads are allowed during this process, but the whole volume must be resynchronized, so there may be some impact on performance of the system, depending on how much data is contained in the volume.

RAID 5 volumes are handled a little differently following a system crash, as it is possible to replay the associated RAID 5 logs if they have been configured. If there are no logs, the volume is placed in *reconstruct-recovery* mode and parity is regenerated. If no action is taken, then there may be corruption of the parity in the RAID 5 volume.

- Dirty region logging (DRL): A log based method that can be employed to aid resynchronization of mirrored volumes. DRL partitions the VxVM volume into regions and keeps track of writes to the volume by region. A single bit is held in a dirty region log on disk to indicate whether a region is dirty or not. Before a write is made, the dirty bit is marked. This is a synchronous operation and must complete before the write is initiated. The dirty bit is cleared but not immediately to allow for multiple writes to the same region without the synchronous write to the DRL log. Log subdisks are used for the DRL log and there can be only one log subdisk per plex. Use of dirty region logging has the greatest impact following a system crash, as only those regions with the dirty bit set need to be resynchronized. This is much faster than the full synchronization of out of sync data. When a plex (copy) has become stale, the whole mirror needs to be resynchronized, so using the DRL will not lengthen the required time. Similar to MWCC, LVM DRL is required to provide mirror consistency after a system crash.
- FastResync: Introduced with Version 3.2 of the VERITAS Volume Manager. FastResync is not enabled by default on creation of new VxVM volumes. FastResync was previously known as Fast Mirror Resynchronization (FMR). The FastResync function keeps track of updates that are missed by a mirror that is unavailable, so that they can be quickly and efficiently synchronized. Where a plex (copy) has become stale, resynchronization with FastResync enabled will be considerably faster. A separate license is required for FastResync.

FastResync types include:

- PersistentResync: Introduced in VERITAS Volume Manager 3.2; holds copies of the FastResync maps on disk, which will persist through a system reboot. This form of FastResync requires a data change object (DCO) and DCO log to be associated with the volume.
- Non-persistent FastResync: Introduced in VERITAS Volume Manager 3.1; holds copies of the FastResync maps in memory. These maps will not persist a reboot but will have little or no impact on I/O performance.

If resynchronization of a mirror copy is executed, without using DRL and/or FastResync, then all data in the plex must be synchronized in response to a manual request. DRL is essential, for FastResync cannot be used for the resynchronization of mirrors after a system crash.

### **Mirroring of the boot disk**

The AIX root file system cannot be encapsulated under VERITAS Foundation Suite for AIX Version 3.4, so mirroring of the boot disk for availability must be carried out by the LVM. The **mirrorvg** command in LVM is provided to mirror the entire root volume group, after it has been created, in a single step.

### **Striping**

Striping, also known as RAID 0, was designed to improve performance by balancing I/O. The basic concept is to split the write portion of an I/O into smaller chunks called stripe units. The stripe units are written to the logical volume in parallel. The read is then also executed in parallel and the data reassembled. Striped volumes alone do not provide redundancy.

Both LVM and VxVM support striping (RAID 0) and mirroring plus striping (RAID 1+0). The difference is that a logical volume is striped in LVM and the plex in VxVM.

In LVM, the stripe characteristics are defined by:

- ▶ stripe size: Also known as the value used to split an I/O into chunks; can be 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, or 128 KB.
- ▶ stripe width: The number of physical volumes that will accommodate the striped logical volume.

In VxVM, the stripe characteristics are defined by:

- ▶ stripe size: Also sometimes referred to in VxVM documentation as the stripe width, it is the value used to split an I/O into chunks. The size is determined by the user, the default is 64 KB.

- ▶ number of columns: As VxVM stripes the plex, the plex must consist of two or more subdisks that reside on different physical disks. Subdisks are then lined up in columns, and these will accommodate the striped plex.

### 6.2.3 Backup

Backup of both data and system configuration is an essential administration task. Data is important and expensive to recreate, if it can be recreated, and growth in storage capacity means that the time required to take backups can be significant. System administrators looked for ways to reduce backup time and recognized the potential that a mirrored environment provided for splitting off a copy of the data for backup purposes. This could be the backup of a raw logical volume or a file system.

Both VERITAS Foundation Suite and AIX LVM and JFS/JFS2 support the use of a copy of data for backup purposes.

In LVM, it is possible to split off a logical volume copy to be mounted at an alternative mount point, as a read only file system, allowing a backup to be taken. Once the backup has completed, the read-only copy is removed by deleting the read-only file system. AIX takes the logical volume copy and resynchronizes it, returning the copy to the mirror pool. Ideally, the file system should be closed or the application quiesced briefly, for the period when the copy is split, to avoid the possibility of there being JFS metadata in flight and not yet written to all mirror copies. Once the split is complete, the application can be restarted. LVM knows that the copy is a split mirror, even through a system crash, and keeps track of partitions in the original mirror that have changed so that only these need to be resynchronized when the split mirror is returned. The split mirror backup can be initiated at the file system level with the **chfs** command and **splitcopy** option. Naturally, to ensure data availability is not threatened, IBM recommends that there are three mirror copies of data to ensure that there are always two active copies while one has been split off for backup purposes.

VxVM encapsulates online backup support via its volume snapshot function in the **vxassist** command. With the snapshot function, it is possible to create a new copy of the data that is called a snapshot mirror with the **snapshot** option, which can then be split off from the original volume. It is also possible to convert an existing mirror into a snapshot mirror. The **snapshot** option of **vxassist** is used to split off a snapshot mirror, and **snapback** is used to return it to the originating volume. A full resynchronization of the mirror would be required unless **FastResync** logging is used. A further option, **snapclear**, can be used to remove the association of the snapshot mirror with the original volume and to create a volume in its own right.

### ***Copy function in hardware***

Alternatively, there are many forms of copy services provided by modern storage subsystems. These include those provided by IBM and other third party vendors, such as EMC and HDS. For example, the IBM Enterprise Storage Server (ESS) has a FlashCopy function and can create a physical point-in-time copy of data. FlashCopy is carried out in such a way that it is possible to access the source and target copies before the copy process has completed. This is still a local copy, so server performance must be considered during the backup process.

### **Online relayout**

Online relayout is a VERITAS term used to describe conversion between volume layouts. A layout might be changed to support different availability and performance requirements. Both LVM and VxVM support the addition and removal of mirror copies on the fly. VxVM supports additional online changes such as modifying the size characteristics of a striped volume and conversion from one type of VxVM volume to another. A full list of permitted relayout transformations and limitations is included in the *VERITAS Volume Manager 3.2 - Administrator's Guide (AIX)*. A temporary area on disk is required to support online relayout, allowing data to be moved in chunks from source to destination. The amount of temporary space for volumes greater than 50 MB is approximately 10% of the volume size, and for volumes less than 50 MB, the temporary space required is the same as the volume size.

### **The mksysb command**

It is essential that the **mksysb** command be considered when running in an AIX environment. The **mksysb** command will back up all mounted file systems in the root volume group. This image can then be used to:

- ▶ Recover the system image, if there is a problem with the system image following a failure or during an upgrade.
- ▶ Clone a system image from one server to another.
- ▶ Provide a bootable image where the mksysb image has been saved to tape or CD-ROM.

It is recommended that a mksysb image be taken prior to any system change. While there is no root encapsulation with VERITAS, the rootvg must be backed up for recovery purposes should the entire system image need to be recovered. This can be done with the **mksysb** command under AIX and can also be achieved through third party products, such as the VERITAS Bare Metal Restore (BMR) product.

### ***Backup of non root disk groups***

On AIX, the **savevg** command can be used to back up non root volume groups and their contents. The volume group structure can be recreated from the **savevg**

archive. Other data backups might be need to be applied to bring the data up to date. Should an entire disk group be lost, this would need to be manually recreated or cloned if appropriate disk group configuration has been previously collected. Backup of data in disk groups can be done manually or automatically through the use of software such as VERITAS NetBackup or Tivoli Storage Manager.

## 6.2.4 Hot spare management

The increased need to run systems continuously, 24 x 7, has led to hardware and software vendors developing functions to automate systems to take action to keep a system up and running in the event of a component failure. One aspect of this is hot spare management. This refers to when the system automatically reacts to I/O failures on redundant LVM or VxVM objects and restores redundancy by relocating the affected object. The system administrator can then take action to replace the failed part without interrupting service. For both LVM and VxVM, hot relocation only makes sense for mirrored environments.

In LVM, hot relocation is carried out at the physical volume level. Specific disks in a volume group can be marked as *hotspare* and the volume group enabled for automated hotsparring. If a disk is determined to go bad or is missing, then LVM will invoke the use of a spare and copy the contents of one physical volume to another. Then, depending on the sync policy, stale partitions will be resynchronized.

In VxVM, subdisks are relocated in response to a disk or plex failure. The *vxrelocd* hot-relocation daemon detects and reacts to these VxVM failures. Subdisks are relocated to disks dedicated as spare disks or those with free space in the disk group if no spares are available. It is also possible to unrellocate subdisks. There are some circumstances where subdisk relocation is not possible; Chapter 9, “Administering Re-location”, of the *VERITAS Volume Manager 3.2 - Administrator's Guide (AIX)* should be consulted for details.

## 6.2.5 JFS/JFS2 and VxFS comparison

IBM's native JFS/JFS2 and VERITAS VxFS are both powerful, high performing, robust, and scalable file systems. They share many similarities, such as journaling with quick recovery capabilities and online defragmentation. In this section, we compare some of the common features and differences between the file systems. For a quick comparison of JFS/JFS2 and VxFS commands and tasks, see the tables in Appendix B, “JFS/JFS2 and VxFS command comparison tables” on page 305.



## JFS and JFS2 comparison

First, let us compare some of the differences between JFS and JFS2. JFS stands for Journaled File System and has been available since AIX Version 3.1. JFS2 is the Enhanced Journaled File System that was introduced on AIX 5L Version 5.1. Both JFS and JFS2 are available on AIX 5L Version 5.1 and later versions. An open source version of AIX's JFS is available for Linux. JFS and JFS2 offer many of the same features, but JFS2 has some enhancements, such as sorted binary-tree (B-tree) directory structures, for faster searches. Table 6-4 compares JFS2 and JFS features.

Table 6-4 Comparison of JFS2 and JFS features

| Feature                               | JFS2                           | JFS                                |
|---------------------------------------|--------------------------------|------------------------------------|
| Fragments/block size                  | 512-4096 block sizes           | 512-4096 fragments                 |
| Maximum file size                     | 4 PB                           | 64 GB                              |
| Maximum file system size              | 4 PB                           | 1 TB                               |
| Number of inodes                      | Dynamic, limited by disk space | Fixed, set at file system creation |
| Inode structure size                  | 512-byte                       | 128-byte                           |
| File space allocation method          | Extent-based                   | Fragment-based                     |
| Directory organization                | B-tree                         | Linear                             |
| Online defragmentation                | Yes                            | Yes                                |
| Large file support                    | Yes, default                   | Yes, mount option                  |
| Compression                           | No                             | Yes                                |
| Direct I/O support                    | Yes                            | Yes                                |
| Default ownership of file at creation | root.system                    | sys.sys                            |
| SGID of default file mode             | SGID=off                       | SGID=on                            |
| Disable journaling                    | No                             | Yes                                |
| Quotas                                | No                             | Yes                                |
| Delayed write support                 | No                             | Yes                                |
| Extended ACL                          | Yes                            | Yes                                |
| Available on POWER architecture       | Yes                            | Yes                                |

| Feature                 | JFS2          | JFS           |
|-------------------------|---------------|---------------|
| Runs on 32-bit hardware | Yes           | Yes           |
| Runs on 64-bit hardware | Yes           | Yes           |
| Kernel optimization     | 64-bit kernel | 32-bit kernel |

### ***Scaling***

A significant advantage between JFS2 and JFS is scaling. JFS2 provides much greater capacity for maximum file sizes and file system sizes. The maximum file size for JFS is 64 GB, but JFS2 supports a maximum file size of 4 PB.

### ***Directory structure***

The use of a sorted binary-tree directory structure greatly speeds up file and directory access in JFS2, as compared to the linear directory structure in JFS.

### ***Delayed writes***

JFS allows delayed writes of data onto disk. The delayed write operations save extra disk operations, because the data is not committed to permanent storage until a process issues the `fsync` command, thus forcing all updated data to be written to disk. This feature is not available with JFS2.

### ***Direct I/O support***

Both JFS and JFS2 offer the Direct I/O caching policy. This caching policy transfers data directly from disk into a user-defined buffer, as opposed to using the normal cache policy of placing pages in kernel memory. Direct I/O therefore offers increased performance.

### ***32-bit and 64-bit kernel optimization***

JFS is optimized for the 32-bit kernel, and will run on 32-bit and 64-bit hardware. JFS2 is optimized for the 64-bit kernel, allowing it to take advantage of 64-bit functionality. JFS2 and JFS are both available on AIX 5L Version 5.1 and later versions. On earlier AIX versions (AIX 3.1 through AIX 4.3.3), only JFS is available.

### ***Coexistence of JFS and JFS2***

Both JFS and JFS2 file systems can coexist easily on the same AIX 5L Version 5.1 machine.

### ***JFS/JFS2 and VxFS comparison***

Obviously, JFS and JFS2 share many commonalities, since both file systems come from the same heritage. But, VxFS also has many features in common with JFS and JFS2. In this section, we will discuss some of the similarities and

differences. There is a quick comparison of some of the features between JFS/JFS2 and VxFS in Table 6-5.

*Table 6-5 JFS/JFS2 and VxFS comparison*

| Feature                               | JFS2/JFS                                  | VxFS                                 |
|---------------------------------------|---|--------------------------------------|
| Block size                            | 512-4096 block sizes (JFS2)               | 1024-8192 block sizes                |
| Maximum file size                     | 4 PB (JFS2)                               | 1 TB                                 |
| Maximum file system size              | 4 PB (JFS2)                               | 1 TB                                 |
| Number of inodes                      | Dynamic, limited by disk space (JFS2)     | Dynamic, limited by global parameter |
| Inode structure size                  | 512-byte (JFS2)                           | 256-byte                             |
| File space allocation method          | Extent-based (JFS2)                       | Extent-based                         |
| Directory organization                | B-tree (JFS2)                             | Hash                                 |
| Online defragmentation                | Yes                                       | Yes                                  |
| Large file support                    | Yes, default                              | Yes, optional                        |
| Compression                           | Yes (JFS); No (JFS2)                      | No                                   |
| Direct I/O support                    | Yes                                       | Yes                                  |
| Default ownership of file at creation | root.system (JFS2)                        | root.system                          |
| SGID of default file mode             | SGID=off                                  |                                      |
| Disable journaling                    | Yes (JFS); No (JFS2)                      | Yes                                  |
| Quotas                                | Yes (JFS); No (JFS2)                      | Yes                                  |
| Delayed write support                 | Yes (JFS); No (JFS2)                      | No                                   |
| Extended ACL                          | Yes                                       | Yes                                  |
| Available on POWER architecture       | Yes                                       | Yes                                  |
| Runs on 32-bit hardware               | Yes                                       | Yes                                  |
| Runs on 64-bit hardware               | Yes                                       | Yes                                  |
| Kernel optimization                   | 64-bit kernel (JFS2); 32-bit kernel (JFS) | 32-bit kernel                        |

## ***Journaling***

Both VxFS and JFS/JFS2 implement journaling. Journaling provides a method of having quicker and better data reliability in the face of a system crash by logging metadata. File system metadata includes, for example, inode, directory, link, and file information.

The file system makes multiple I/O accesses to write metadata changes, and a crash can happen between writing these I/Os to disk. If the crash happens between these I/Os, the file system is not in a consistent state. A non-journaled file system would have to examine all of the file system's metadata using the **fsck** command, which is time-consuming. A journaled file system, though, keeps track of any file system metadata changes in a log before they are written. If there is a system crash, then the log can be replayed to insure file system consistency.

VxFS calls this log of metadata an intent log, and JFS/JFS2 calls it a journal log, but they are equivalent. The VxFS intent log is a circular log, and the size of the intent log can be set by the administrator at the time of file system creation. The default size of the intent log is 1024 bytes with a maximum size of 8192 bytes. It is possible to disable the intent logging with VxFS, but this is not recommended.

With JFS/JFS2, the journal log is also a circular log, but the size of the journal log is automatically allocated when the file system is created. The default journal log size is 4 MB, with a maximum of 256 MB. Like VxFS, JFS allows you to disable journal logging, but JFS2 does not.

By default, JFS and JFS2 use one common journal log for multiple file systems. The log is located on a separate logical volume. In contrast, VxFS requires a single intent log per file system. It is possible for the log to be either internal or external for both JFS/JFS2 and VxFS. An internal log means that the metadata is in the same physical location as the file system. An external log means that the log is located in a separate physical location from the file system. Using an external log helps improve I/O throughput, since writes to the file system and the log can be done in parallel. By default, JFS and JFS2 use external logs, but it is possible to specify an internal (or inline) log with JFS2. VxFS is different. The intent log in VxFS is, by default, at the beginning of a volume. Performance may be improved by making the intent log external and moving it off to a separate physical location. The VERITAS QuickLog product allows you to do this.

## ***Inodes***

Inodes contain access information for files as well as pointers to the real disk addresses of the file's data blocks. There is one inode per file.

The inode in JFS2 is a 512 byte structure. Each inode includes the information shown in the first column of Table 6-6 on page 213.

The number of inodes is dependent on the size of the file system and are allocated dynamically as needed. The number of available inodes can be determined by using the **df -v** command. Inodes are defined in the `/usr/include/jfs/ino.h` file.

JFS is similar but has some differences. The number of inodes are fixed and set at file system creation, instead of being dynamically created as with JFS2. Also, each inode structure is 128 bytes versus the 256 bytes for JFS2. The content of the inode structure is also slightly different. See the second column of Table 6-6 for the JFS inode structure and how it compares to the JFS2 inode structure.

*Table 6-6 JFS and JFS2 inode structures*

| <b>JFS2 Field</b> | <b>JFS Field</b> | <b>Contents</b>   |
|-------------------|------------------|---|
| di_mode           | i_mode           | Type of file and access permission mode bits              |
| di_size           | i_size           | Size of file in bytes                                     |
| di_uid            | i_uid            | Access permissions for the user ID                        |
| di_gid            | i_gid            | Access permissions for the group ID                       |
| di_nblocks        | i_nblocks        | Number of blocks allocated to the file                    |
| di_mtime          | i_mtime          | Last time file was modified                               |
| di_atime          | i_atime          | Last time file was accessed                               |
| di_ctime          | i_ctime          | Last time inode was modified                              |
| di_nlink          | i_nlink          | Number of hard links to the file                          |
| di_btroot         |                  | Root of B+ tree describing the disk addresses of the data |
|                   | i_rdaddr[8]      | Real disk addresses of the data                           |
|                   | i_rindirect      | Real disk address of the indirect block, if any           |

VxFS shares similarities with JFS2. Just like with JFS2, in VxFS inodes are allocated dynamically as they are needed. The maximum number of inodes, though, can be limited by setting a global parameter that differs from JFS2. Like JFS2, the default inode size is 256 bytes, but unlike JFS2, the inode size can be set when the file system is created. Although a VxFS inode is typically 256 bytes in length, an inode size of 512 bytes can be used instead. Again, the default size (256 bytes) is recommended. VxFS inode structures contain similar information to JFS2, although there are more fields. See Table 6-7 for the VxFS inode structure.

*Table 6-7 VxFS inode structure*

| Field      | Contents   |
|------------|--|
| i_mode     | The mode and type of file  |
| i_nlink    | The number of links to the file  |
| i_uid      | The inode owner  |
| i_gid      | The inode group  |
| i_size     | The size in bytes of the file  |
| i_atime    | Last time file was accessed (timestruct_t format)  |
| i_mtime    | Last time file was modified (timestruct_t format)  |
| i_ctime    | Last time inode was modified (timestruct_t format)   |
| i_aflags   | Flags that control the allocation and extension of files   |
| i_orgtype  | Indicates how the inode mapping area is to be interpreted  |
| i_eopflags | Extended inode operation flag area   |
| i_eopdata  | Extended inode operation data area   |
| i_ftarea   | A union containing device, directory, or file information  |
| i_blocks   | Number of blocks allocated to the file, including blocks allocated for indirect address extents                      |
| i_gen      | Generation number provided as a "handle" for stateless servers such as NFS   |
| i_vversion | Count of the number of times the inode metadata has been modified  |
| ic_org     | Union containing mapping area information  |
| i_iattrino | Indirect attribute inode. Identifies the inode in the attribute fileset that contains indirect attribute references. |

The remaining bytes of the VxFS inode can contain extended attribute record information. When attributes are too large to be stored directly in the inode, each attribute is stored in its own file. This record lists each attribute along with the inode number corresponding to the file in which the attribute is stored. As we can clearly see, the VxFS inode contains much more information in the inode structure than JFS or JFS2.

### ***Online file system resizing***

Quite often over time, a file system becomes too small to contain the growth of files. It is handy to be able to increase the file system size as your data storage needs increase. Both VxFS and JFS/JFS2 offer the ability to grow the size of a file system online without any disruption to users logged in. The file system is increased transparently while users are on the system, including those who may be accessing the file system whose size you are increasing. As long as the physical volume or device is big enough to hold the increased size of the file system, it is a simple matter to increase the size with either VxFS or JFS/JFS2.

With VxFS, you use the **fsadm** command, which will allow you to increase the file system in 512-byte blocks. Again, remember that the file system can only be increased up to the size of the physical device that the file system resides on.

With JFS and JFS2, you can use either the **chfs** command or the smitty interface to increase the file system size. The smitty fast path to changing the file system characteristics (including the file system size) is **smitty chfs**. Just like with VxFS, you can increase the file system in 512-byte blocks. If you place a '+' in front of the number, then the request is increased by that exact specified amount. But, if the specified size is not evenly divisible by the physical partition size, it is rounded up to the closest number that is evenly divisible. In JFS/JFS2, the volume group in which the file system resides defines a maximum logical volume size and limits the file system size.

Although it is more common that a file system will need to be increased, it is possible that you may have overestimated the file system size and find that you want to decrease its size. VxFS offers the ability to decrease or shrink a file system. At the time of writing, JFS and JFS2 do not offer this capability. The VxFS **fsadm** command is used to shrink a file system, just like with growing a file system. Since VxFS uses a truncation method to decrease the file system size, it may not be possible to decrease file systems where data is allocated at the end of the file system.

### ***Online defragmentation***

If a file system has become fragmented, there may not be large enough fragments to allocate. Defragmenting the file system is then useful to reorganize the file system data and create a larger section of contiguous free space. Both VxFS and JFS/JFS2 allow you to defragment a file system online, even while the

file system is mounted and users are accessing it. VxFS uses the **fsadm** command to defragment a file system. It is recommended to run **fsadm** periodically to keep file system performance optimal. Depending on file system activity, defragmentation should be done between once a day and once a month. **fsadm** has several options to control the defragmentation process. One option, **-t**, sets a limit on how long the defragmentation will run. Another option, **-p**, limits the number of passes done to do the defragmentation. The default number of passes is 5. Directories and extents (file system data) can be specified separately to be defragmented. A report can also be generated via the **-v** or **-D** options. JFS and JFS2 use the **defragfs** command, which accomplishes the same defragmentation task. Like the VxFS **fsadm** command, **defragfs** has the option to report on the defragmentation through the **-q** or **-r** options. JFS and JFS2 do not offer options to set a time limit or a limit on the number of passes when doing the defragmentation.

### ***JFS compressed file systems***

Unlike JFS2 and VxFS, JFS supports compressed file systems. The allocation of disk space for compressed file systems is the same as that of fragments in fragmented file systems for JFS or JFS2. Each logical block is allocated 4096 bytes when it is modified. After the modification is complete, then the logical block is compressed before it is written to a disk. The compressed logical block is then allocated only the number of fragments required for its storage.

### ***Caching***

As discussed in Chapter 2, “Components” on page 9, VxFS provides cache advisories that allow an application to set how I/O is performed and files are accessed. To set a cache advisory, you use the **ioct1** command. This section discusses the differences between VxFS caching advisories and JFS and JFS2 disk I/O protocols.

VxFS has one cache advisory called direct I/O, which is an unbuffered form of I/O, in that I/O bypasses the kernel buffer and goes directly to a user-defined buffer. As described in Chapter 2, “Components” on page 9, with VxFS, direct I/O CPU usage is equivalent to doing a raw disk transfer because of decreased CPU overhead, but in certain circumstances, the performance of normal, buffered I/O may be faster than direct I/O, depending upon the type of I/O request. This is similar to the difference between direct I/O and normal, cached I/O for JFS and JFS2.

Direct I/O is also supported on JFS and JFS2. It works slightly differently, but has the same ability to provide an application with the ability to have a type of unbuffered I/O. When you are processing normal I/O to JFS or JFS2 files, the I/O goes from the application buffer to the kernel's Virtual Memory Manager (VMM) and from there to the user buffer. This normal I/O processing actually is a type of cached I/O; it makes use of a file buffer cache which the VMM uses. If the file



cache hit rate is high and well-utilized, then this normal, cached I/O will improve I/O performance. If, on the other hand, the cache hit rate is low, or an application has very large I/O requests, then the normal cached I/O is not that useful. In that case, an application can use direct I/O.

Direct I/O with JFS and JFS2 eliminates the copy from the VMM file buffer cache to the user-defined buffer, which is the same as the bypassing of the kernel buffer that VxFS does with direct I/O. To use direct I/O, a file can be opened with the `O_DIRECT` option.

The performance of direct I/O with JFS and JFS2 is about the same as raw I/O, but raw I/O is still slightly faster, depending on the type of I/O requests.

Normal, cached I/O may also be faster than direct I/O in some circumstances. This is because VMM actually uses a read-ahead algorithm as well as the file buffer cache. The read-ahead algorithm is very useful for sequential access to files because the VMM can initiate disk requests and have the pages already be resident in memory before the application has requested the pages. Applications can compensate for the loss of this read-ahead by either issuing larger read requests (minimum of 128 KB), or using asynchronous direct I/O. Also, when doing I/O writes, direct I/O can have a significant performance penalty because a write-behind algorithm is bypassed. The write-behind policy that is used with normal cached I/O has writes that can go to memory and then be flushed to disk later. So applications that do a lot of read/writes would benefit from normal, cached I/O because of the read-ahead and write-behind algorithms that are used.

VxFS also offers discovered direct I/O is similar to direct I/O, but makes use of a parameter, `discovered_direct_iosz`. This parameter says that if an I/O request is larger than the parameter, then direct I/O is used for that request. JFS and JFS2 do not offer a similar parameter.

Asynchronous I/O is another type of I/O protocol offered by JFS/JFS2. This is not available with VxFS. In asynchronous I/O, the I/O can run in the background and does not block user applications. This is in contrast to normal I/O processing where an application's processing cannot continue until the I/O operation is complete. Performance may be improved by using asynchronous I/O because I/O operations and application processing can run in parallel. An application can monitor the asynchronous I/O request by either polling for the status of the I/O request, or the system can asynchronously notify the application when the I/O request has completed. The application can, of course, block until the I/O request has completed, if necessary. To enable asynchronous I/O, use the `smitty aio` fast path.

Both VxFS and JFS/JFS2 offer the ability to do synchronous I/O, but VxFS also offers another cache advisory called data synchronous I/O. With data

synchronous I/O, the data is transferred to disk synchronously before the write returns to the user. The data is written, as well as the inode if there is a file size increase.

### ***Coexistence of JFS/JFS2 and VxFS***

VxFS and JFS/JFS2 file systems can all coexist without any problem on the same AIX 5L machine. For example, you can have some native AIX logical volumes containing JFS or JFS2 file systems on a given AIX 5L machine, and you can have some VxVM volumes containing VxFS file systems on that same AIX 5L machine. Note that it is not possible to have a JFS or JFS2 file system on a VxVM volume, nor is it possible to have a VxFS file system on a native AIX LVM logical volume. There is no technical reason why this is not possible, it is simply not supported. All three types of file systems, VxFS, JFS, and JFS2, can be administered through SMIT (or smitty). You can also use the respective command-line interface (CLI) commands for all three file systems. Lastly, VxFS can be administered through the VEA GUI, but VEA does not support administering of the JFS or JFS2 file systems.

## Migration considerations

This chapter describes some of the migration or integration scenarios that might be encountered as a business considers implementing storage management using VERITAS Foundation Suite for AIX.

Migration topics covered include:

- ▶ A migration from SUN Solaris to AIX
- ▶ A migration from AIX LVM and JFS to VxVM and VxFS
- ▶ Useful tools and practical methodologies for migration

## 7.1 Reasons for migration

This section reviews some of the reasons for migrating from an existing environment to a platform running VERITAS Foundation Suite for AIX.

A migration might be undertaken for the following reasons:

- To take advantage of IBM @server pSeries hardware: pSeries UNIX servers are designed to deliver the highest levels of reliability, performance, and flexibility.

Today's business environment dictates that the IT infrastructure is mission-critical, which means that unscheduled downtime is not an option. At the heart of every pSeries component, reliability plays a major role. The POWER4 processor, used in the latest pSeries servers, such as the p690, p670, and the p630, contains system sensors that are constantly monitoring for errors and failures. This allows components to be deallocated without interruption to the operating system and ensures that when a failure does occur that the failed component is always identified. The pSeries servers also use Chipkill memory with built-in spare memory modules, which means that even in the event of a total memory chip failure, the server can keep running. Unique packaging technologies used in the mainframe are also deployed in pSeries to ensure continuous availability. These and many other IBM technologies ensure the most reliable UNIX servers available.

The pSeries servers are also designed to deliver the highest levels of performance possible. This is achieved through advanced systems design and the use of powerful processors. The latest processor, the POWER4, is the first *system on a chip* and packages two microprocessors together with a Level 2 cache and system interconnects on a single piece of silicon. This allows the chip to operate at very high speeds and to scale efficiently to a high number of processors.

- To consolidate server workload: The last decade has seen the number of servers installed at customer sites increase to immense proportions. As new applications and functions were added to the IT infrastructure, these were usually installed on separate individual servers. This has meant that in many environments, there can be literally thousands of servers installed. Managing these servers and their growth has become too costly and complex. Server consolidation offers a way to reduce the number of servers and the complexity of managing these servers by reducing the total number of systems installed. Utilizing tools such as Workload Manager (part of the base AIX operating system) allows multiple applications to be installed on a single pSeries server while managing each applications' individual resource requirements. Logical partitioning goes one step further and allows multiple applications to be installed on a single pSeries server while isolating applications from each other by running under separate copies of AIX but

sharing one common server. The pSeries servers are ideal server consolidation platforms because they offer the ability to host multiple applications due to the high levels of reliability (since any single system failures will affect all applications), the high levels of performance, and the flexibility of the logical partitioning to adapt to different application resource requirements.

- ▶ To provide for a flexible environment for future growth: Flexibility is a key part of the pSeries servers and is delivered through the operating system and the logical partitioning technologies that have been brought from the mainframe. Logical partitions allow up to 16 individual copies of the AIX operating system to be running on a single server. The POWER4 based servers are able to partition the system down to a single CPU, a single I/O adapter, and a section of memory (256 MB chunks). The fine levels of granularity mean that the system is not bound by physical resources, such as system boards, and partitions can easily be expanded or contracted by these individual processors/adapter or memory sections.
- ▶ To leverage additional products available from VERITAS that are built upon VERITAS Foundation Suite: In the competitive commercial environment that exists today, most organizations require not just storage management and data protection but also increased levels of availability. The VERITAS product suite includes components to support high availability and disaster recovery, such as VERITAS Cluster Server and VERITAS Volume Replicator. Use of VERITAS products, which are tightly integrated with a common look and feel, can make migration to VERITAS Foundation Suite for AIX a desirable option. Another key factor is support for management of heterogeneous environments with VERITAS Foundation Suite for AIX.

**Note:** VERITAS Cluster Server (VCS) can be used with AIX LVM and JFS/JFS2 or VxVM and VxFS.

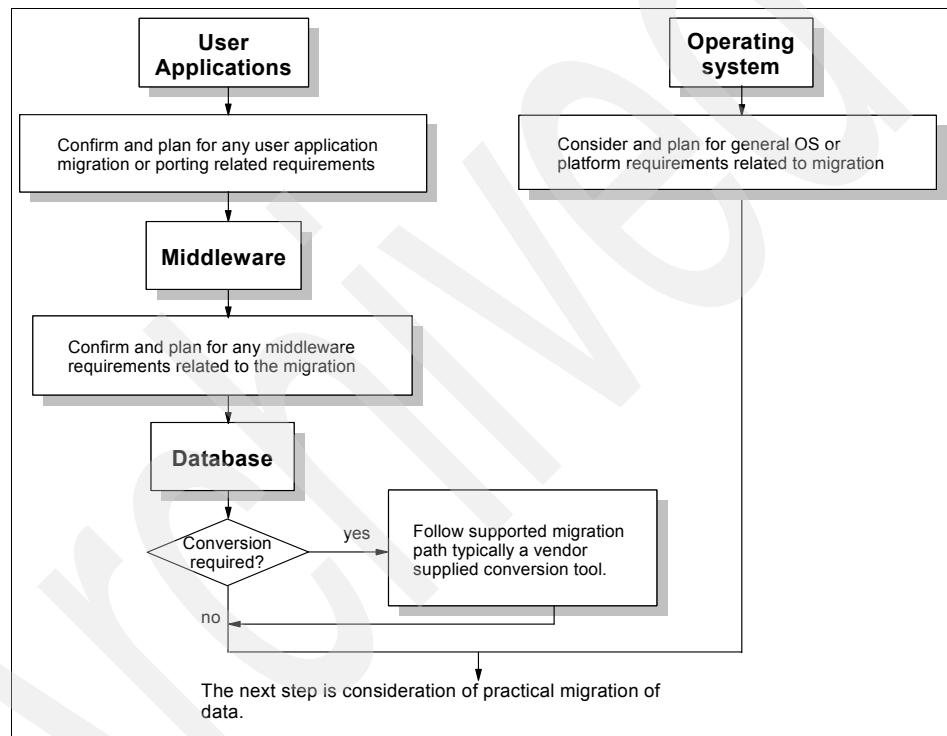
The rest of this chapter concentrates on migration planning and processes. The two main areas examined are: planning for migration at the application and operating system level, and the practical requirements related to storage migration and management. It is not the intent of this chapter to cover migration of an application or other software with examples; however, the issues involved will be considered.

## 7.2 Planning for migration

It is clear that for most businesses a migration to, or introduction of, new hardware or software requires prior planning. The reasons for making a change must be well understood and be valid for the business now and in the future.

Once this has been confirmed, the responsibility moves to the implementers of the solution. In a large organization, this may consist of a number of different groups, each responsible for separate aspects of the environment, such as applications, backups, or the operating system. In this case, good communication will also be essential.

Planning for migration should begin at the software level. Figure 7-1 gives a high level overview of this process. It is also appropriate to review and plan for any operating system related requirements, especially if there is a migration to another platform.



*Figure 7-1 Planning for migration of application and the operating system*

Following migration planning for the existing software and applications, the next step is to consider the practical migration of data. Figure 7-2 on page 223 describes the key migration questions and options when planning for migration of the storage.

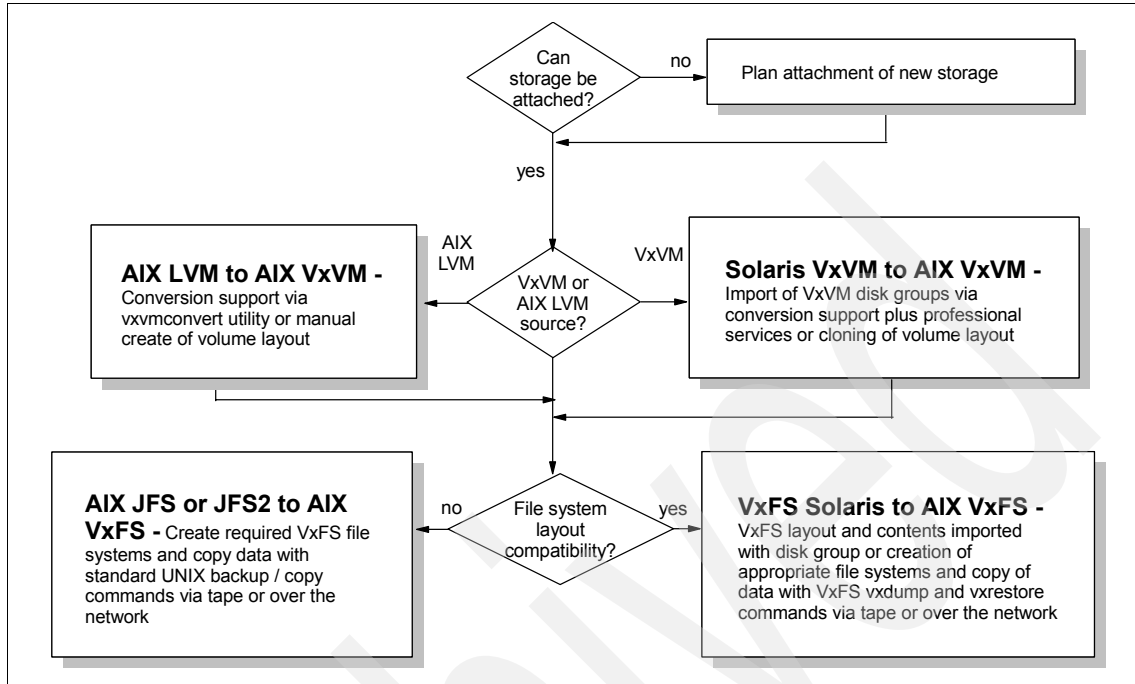


Figure 7-2 Planning for practical migration of data

## 7.2.1 Applications

Availability of an application on the server platform is probably of equal importance to availability of data in the environment. It is the use of the application that generates income for the organization and enables day to day business to take place. Once it has been confirmed that the application is compatible with the given hardware and operating system level, one of the most useful strategies can be to begin application testing on the new platform at an early stage in the project. Indeed, it might be appropriate to obtain a loan system for doing testing if suitable test systems are not available locally. IBM offers the Solution Partnership Centers (SPCs), which are located worldwide. The SPCs help a business to test their applications on all IBM platforms, including IBM @server pSeries, and providing expert technical assistance. More information on the SPCs can be found at

<http://www.developer.ibm.com/spc/>

One question you might have is whether to make the data available in the same way on pSeries as compared with a Sun or another UNIX platform. Essentially, if the application is using raw volumes in one environment, it should be suitable to use raw volumes on the new platform and similarly for file systems. However,

there are some caveats, which we note at the end of this section. If the application data structures are different between platforms, then further steps may be required to complete the migration from an application perspective.

Tuning on each platform will be application dependent, but typically on AIX, the dynamic nature of the kernel means that less parameters have to be explicitly set. It would be appropriate to talk with your application vendor about these details.

In parallel with testing the application on the new platform, it is important to decide how the final move to a production environment will be made. Will the systems be run in parallel first or will there be a single cut-over point? In addition, existing backup strategies will need to be adjusted or amended to support the move. These decisions are business dependent and must be made at a company level.

**Note:** It is essential that the application vendor be approached prior to any platform change to confirm supported migration paths. This includes database vendors, as conversion of data structures between platforms is typically required.

## 7.2.2 Operating system considerations

As we see in Figure 7-1 on page 222, part of the migration planning process requires thoughtful planning of the base operating system migration. Part of this migration planning is the practical challenge for the system administrator in managing a new and different UNIX operating system. System administration tasks, such as user management, device management, and fix updates, are very much the same across different platforms, and IBM has a number of easily accessible resources to enable the system administrator to make a smooth transition from any UNIX platform to AIX.

These resources include:

- ▶ **Solaris Affinity:** A feature of AIX 5L that is comprised of a number of extensions to AIX. The aim is to assist current Solaris administrators to be immediately productive on AIX. Key features of Solaris Affinity are SVR4 based commands and subsystems, such as **prtconf**, **truss**, and the **/proc** file system.
- ▶ **Linux affinity:** Linux affinity is an integral part of AIX 5L, where the 'L' stands for Linux. The Linux affinity includes binary compatibility with Linux binaries, and also extends to include the Linux Toolbox for AIX, which consists of over 300 pre-built, open source tools. Go to



<http://www.ibm.com/servers/aix/products/aixos/linux/index.html> for a full listing of available tools.

- ▶ System Management Interface Tool (SMIT): An easy-to-use and intuitive user interface that can be used for nearly all administration tasks on AIX. The SMIT interface allows an experienced UNIX administrator to function productively almost at once with a menu-based system that is straightforward to navigate. There is no requirement to know in detail the underlying commands to carry out any given task or specifically how AIX implements a function.
- ▶ IBM Learning Services (ILS): ILS offers a conversion to AIX course for experienced UNIX system administrators titled “AIX Jumpstart for UNIX professionals”. Visit the ILS Web site at <http://www.ibm.com/services/learning/> for a course outline.
- ▶ IBM Redbooks: IBM Redbooks, such as this one, are the output of intensive multi-week work efforts, where a small team of people explore and document product implementation, integration, and operation. Redbooks exist across the whole @server product range for both hardware and software. An example of a redbook is *AIX Reference for Sun Solaris System Administrators*, SG24-6584. See <http://www.redbooks.ibm.com> for a complete list of available redbooks.

## The user environment

If an existing environment is being moved or consolidated, a further consideration will be how to maintain the user community. For many databases, there will be only a small number of users to migrate, with many real end users connecting to the database via a single ID. In this case, the process of installing the application will often create all the required user IDs.

Where there are a large number of individual user IDs to migrate, then it is useful to understand how the user information is stored on one platform compared with another. Take Solaris to AIX as an example. Comparable user related files from the two platforms are documented in Table 7-1.

Table 7-1 User configuration files on IBM AIX compared with Sun Solaris

| Task/file locations               | AIX 5L Version 5.1                  | Solaris 8                  |
|-----------------------------------|-------------------------------------|----------------------------|
| Password files                    | /etc/passwd<br>/etc/security/passwd | /etc/passwd<br>/etc/shadow |
| Group files                       | /etc/group<br>/etc/security/group   | /etc/group                 |
| Process resource limits for users | /etc/security/limits                | N/A                        |

| Task/file locations             | AIX 5L Version 5.1               | Solaris 8               |
|---------------------------------|----------------------------------|-------------------------|
| System wide environment file    | /etc/environment<br>/etc/profile | /etc/environment        |
| Defaults user profile file      | /etc/security/.profile           | /etc/skel/local.profile |
| Default user configuration file | /usr/lib/security/mkuser.default | N/A                     |

The default user configuration file on AIX, `mkuser.default`, which is sourced by the `mkuser` command, can be edited to include common user characteristics. The migration of the required user community could then be scripted, possibly taking input from the user files on the originating system. For a user community numbered in the high thousands, the task of migration can be undertaken as part of a services engagement. When the majority of the user community is handled by some form of directory service based on the lightweight directory protocol (LDAP), then migration is no longer considered at the operating system level. LDAP is a standard protocol supported by AIX, so interoperability is made easier by this fact. One advantage in using an LDAP directory server, such as the IBM SecureWay Directory, is the ability to externalize authentication, minimizing the amount of user ID and password management handled by individual servers.

The next consideration is practical migration of data. IT consultants with experience with VERITAS Foundation Suite on another UNIX platform will find management of storage looks and feels very much the same on AIX 5L. The practical examples described in this redbook have been tested in a lab environment and can be considered valid as a proof of concept exercise. As with all UNIX environments, physical layout of data is important, and is still the responsibility of the system administrator. Often, best practices will be made available by a specific vendor to meet the requirements of their application, which can then be applied to creation of VxVM objects.

## 7.3 Migration of VxVM and VxFS on Solaris to AIX

When migrating data from an existing environment, work will have already been done to design a storage layout that meets known performance and availability requirements. Ideally, we want to be able to reuse existing configuration information when implementing the storage layout on the target platform to minimize the practical effort involved in migration. Of course, if the existing layout can be improved, then this is a good opportunity to make changes, but additional time will be required for planning and implementing of the new layout. The first area we will look at, after a brief description of the test environment, is cloning of volume layouts.

### 7.3.1 Test environment configuration

The source system is a Sun Ultra 60 running Solaris 8 and VERITAS Foundation Suite for Solaris. This system has two internal disks, neither of which are partitioned. The root file system is encapsulated under VERITAS Foundation Suite for Solaris. On Sun Solaris, encapsulation of the root file system is usually done to enable mirroring of the boot disk. In this test environment, the boot disk was encapsulated to enable the creation of a separate disk group for testing purposes. The initial disk configuration can be seen in the command output in Example 7-1.

*Example 7-1 Initial disk configuration on Sun Ultra 60 test system*

---

```
[On Solaris system]
# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
  0. c0t0d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
    /pci@1f,4000/scsi@3/sd@0,0
  1. c0t1d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
    /pci@1f,4000/scsi@3/sd@1,0
# vxdg list
NAME      STATE      ID
rootdg    enabled    1027954259.1025.sunb
testdg    enabled    1028044445.1084.sunb
```

---

Two VxFS file systems were created on the Solaris system. One that contained some data to be migrated; the other was a container for backups. In this case, /test was populated with some JPEG (.jpg) images of RS/6000 and pSeries servers that could be viewed from a Web browser. /test/testdir contained a simple test ksh script.

To create the required file systems, run:

```
# vxassist make testvol 1g testdg01
# mkfs -F vxfs /dev/vx/rdisk/testdg/testvol 4096
# mkdir /test
# mkdir /test/testdir
# mount -F vxfs -o log /dev/vx/dsk/testdg/testvol /test

# vxassist make backupvol 1g testdg01
# mkfs -F vxfs /dev/vx/rdisk/testdg/backupvol 20048
# mkdir /backup
# mount -F vxfs -o log /dev/vx/dsk/testdg/backupvol /backup
```

The target system is a stand-alone RS/6000 F80 running AIX 5L Version 5.1, maintenance level 02, and VERITAS Foundation Suite for AIX. The server has a 32-bit kernel, a JFS file system, and 1 GB of memory. The F80 has multiple

internal disks, and as there is no encapsulation of file systems in rootvg, a separate rootdg is seen. All the disks other than those used for rootvg and rootdg are available for use in the migration.

*Example 7-2 Initial disk configuration on IBM F80 test system*

---

```
[On AIX system]
# lspv
hdisk0          0001615f43bd8180          rootvg
hdisk1          0001615fcbcl83f          VeritasVolumes
hdisk2          0001615fcbcl86b          VeritasVolumes
hdisk3          0001615fcbea5d16          VeritasVolumes
hdisk4          0001615fcbea5e58          VeritasVolumes
hdisk5          0001615fcbea5f96          None
# vxdg list
NAME            STATE            ID
rootdg          enabled         1027642829.1025.srvr80e
```

---

VERITAS Foundation Suite Version 3.4 was installed on both the AIX and Solaris test servers. This software includes VERITAS Volume Manager (VxVM) 3.2 and VERITAS File System 3.4.

## 7.3.2 Cloning volume layouts

It is common practice for a system administrator to use the VxVM commands **vxprint**, **vxdisk**, and **vxprivutil** to capture a description of the current VxVM environment. The output from these commands is typically gathered for backup purposes and can be used to recreate an existing VxVM setup if, for example, there is corruption following a system problem. For migration purposes, the aim is to use some of this information to clone existing layouts.

Detailed below is one possible subset of VxVM commands that could be used for backup purposes. Note that some of the commands are executed at the disk group level, while others are executed at the device level. The **vxprivutil** command is provided for diagnostics purposes only.

On the AIX or Solaris server, you can do the following; if desired, these commands could be placed in a script and run at set intervals by using cron:

1. Create a local save directory that will not be lost on reboot.

```
# mkdir /var/tmp/veritas_save
# cd /var/tmp/veritas_save
```

2. Capture a list of configured VM disks:

```
# /usr/sbin/vxdisk list > vxdisk-list
```

3. Back up VxVM configuration information at the disk group level for each disk group known to the system:

```
# /usr/sbin/vxprint -htg dname > vxprint-thg.dname
# /usr/sbin/vxprint -g dname -mpvshr > vxprint-gmpvshr.dname
# /usr/sbin/vxprint -qQvshpmg disk_group > vxprint-qQvshpmg.dname
```

4. Back up VxVM configuration information at the disk device level:

```
# /usr/lib/vxvm/diag.d/vxprivutil dumpconfig disk_device_path > \
vxprivutil-dumpconfig
# /usr/lib/vxvm/diag.d/vxprivutil scan disk_device_path > vxprivutil-scan
# /usr/lib/vxvm/diag.d/vxprivutil list disk_device_path > vxprivutil-list
```

At the disk group level, there are differences on Solaris compared with AIX, mainly in the header information used on each platform. This means that we cannot directly use the disk group information from Solaris to recreate disk groups on AIX. However, creation of disk groups is very simple as can be seen in the practical example that follows in “Example of cloning volume layouts” on page 230. Device naming conventions also differ significantly between Solaris and AIX.

The output from **vxdisk list** on both the test configuration boxes, for Solaris and AIX, is shown in Example 7-3, which highlights the difference in device naming convention. The output from **vxdg list** on the disk group shows the header information on each platform.

*Example 7-3 Device naming conventions and disk group headers AIX/Solaris*

---

```
[On Solaris system]
# vxdisk list
DEVICE      TYPE      DISK      GROUP      STATUS
c0t0d0s2    sliced    disk01    rootdg     online
c0t1d0s2    sliced    testdg01  testdg     online nohotuse
# vxdg list testdg
Group:      testdg
dgid:       1028044445.1084.sunb
import-id:  0.1083
flags:
version:    90
detach-policy: global
copies:     nconfig=default nlog=default
config:     seqno=0.1042 permten=1570 free=1565 templen=3 loglen=238
config disk c0t1d0s2 copy 1 len=1570 state=clean online
log disk c0t1d0s2 copy 1 len=238

[On AIX system]
# vxdisk list
DEVICE      TYPE      DISK      GROUP      STATUS
hdisk0      simple    -         -          LVM
```

```

hdisk1      simple - - online
hdisk2      simple disk01 rootdg online
hdisk3      simple - - online
hdisk4      simple testdg01 testdg online
hdisk5      simple - - LVM
# vxdg list testdg
Group:      testdg
dgid:       1029853213.1204.srvr80e
import-id:  0.1203
flags:
version:    90
detach-policy: global
copies:     nconfig=default nlog=default
config:     seqno=0.1027 permlen=3004 free=3002 templen=2 loglen=455
config disk hdisk4 copy 1 len=3004 state=clean online
log disk hdisk4 copy 1 len=455

```

---

The steps outlined “Example of cloning volume layouts” on page 230 show how it is possible to capture the volume structure from a Sun Solaris system and to apply it to an AIX system to achieve the same volume layout.

### Example of cloning volume layouts

The **vxprint** output in Example 7-4 shows the current disk group and volume layout on the originating Sun Ultra 60 that we will clone to the IBM RS/6000 F80. The recreation step uses the **vxmake** command, which can take configuration information from a file.

*Example 7-4 Volume layout on source Sun Solaris system to be cloned*

---

```

[On Solaris system]
# vxprint -g testdg

```

| TY | NAME         | ASSOC        | KSTATE  | LENGTH  | PLOFFS | STATE    | TUTILO | PUTILO |
|----|--------------|--------------|---------|---------|--------|----------|--------|--------|
| dg | testdg       | testdg       | -       | -       | -      | -        | -      | -      |
| dm | testdg01     | c0t1d0s2     | -       | 8376480 | -      | NOHOTUSE | -      | -      |
| v  | backupvol    | fsgen        | ENABLED | 2097152 | -      | ACTIVE   | -      | -      |
| pl | backupvol-01 | backupvol    | ENABLED | 2097360 | -      | ACTIVE   | -      | -      |
| sd | testdg01-02  | backupvol-01 | ENABLED | 2097360 | 0      | -        | -      | -      |
| v  | testvol      | fsgen        | ENABLED | 2097152 | -      | ACTIVE   | -      | -      |
| pl | testvol-01   | testvol      | ENABLED | 2097360 | -      | ACTIVE   | -      | -      |
| sd | testdg01-01  | testvol-01   | ENABLED | 2097360 | 0      | -        | -      | -      |

---

On the Solaris system:

1. Capture the current volume information for the disk group:

```
# /usr/sbin/vxprint -g testdg -mpsvh > vxprint-gmpsvh.testdg
```

2. Transfer the **vxprint** output file to the target AIX server via ftp:

```
# ftp srvr80e
(login as root)
ftp> bin
ftp> put /var/tmp/veritas_save/vxprint-gmpvsh.testdg
```

On the AIX system:

1. Create a new disk group and assign a VxVM disk to it. The disk group and VxVM disk should have the same disk name used on the Sun system. This step assumes that **hdisk4** has already been placed under the control of VxVM and is available for use in a disk group. Run **vx dg list** to check that the disk group has been created:

```
# lspv hdisk4
# vx dg init testdg testdg01=hdisk4
# vx dg list
```

2. In this case, we have only one disk in the disk group; to add additional disks to the disk group, run the following command, substituting the appropriate values for disk and device:

```
# vx dg -g testdg adddisk disk=device
```

3. Two small edits to the **vxprint** file from the Solaris system are required before it can be used with the **vxmake** command on the AIX server. Modify each occurrence of **group=root** in this file to be **group=system**. On Solaris, the root user has a group root, and on AIX, the root user's group is system, so we need to make a change:

```
# vi /var/tmp/veritas_save/vxprint-gmpsvh.testdg
...
minor=6001
user=root
group=root change all occurrences to group=system
mode=0600
....
:wq /var/tmp/veritas_save/vxprint-edit.testdg
```

4. Run **vxmake** with the modified **vxprint** output. After **vxmake** has been executed, run **vxprint** to view the disk group and newly created VxVM objects:

```
# vxmake -g testdg -d /var/tmp/veritas_save/vxprint-edit.testdg
# vxprint -g testdg
```

At this stage, each volume is KSTATE DISABLED and LENGTH EMPTY, so **vxvol** must be run to start each individual volume. Use of **vxvol** with the **startall** option will not work in this scenario:

```
# vxvol start testvol
# vxvol start backupvol
```

See Example 7-5 for the complete output from **vxprint** after executing all the commands in this step.

*Example 7-5 Volume layout on AIX system after cloning*

|                     |              |         |          |        |        |        |        |
|---------------------|--------------|---------|----------|--------|--------|--------|--------|
| [On AIX system]     |              |         |          |        |        |        |        |
| # vxprint -g testdg |              |         |          |        |        |        |        |
| TY NAME             | ASSOC        | KSTATE  | LENGTH   | PLOFFS | STATE  | TUTILO | PUTILO |
| dg testdg           | testdg       | -       | -        | -      | -      | -      | -      |
| dm testdg01         | hdisk4       | -       | 17769808 | -      | -      | -      | -      |
| v backupvol         | fsgen        | ENABLED | 2097152  | -      | ACTIVE | -      | -      |
| pl backupvol-01     | backupvol    | ENABLED | 2097360  | -      | ACTIVE | -      | -      |
| sd testdg01-02      | backupvol-01 | ENABLED | 2097360  | 0      | -      | -      | -      |
| v testvol           | fsgen        | ENABLED | 2097152  | -      | ACTIVE | -      | -      |
| pl testvol-01       | testvol      | ENABLED | 2097360  | -      | ACTIVE | -      | -      |
| sd testdg01-01      | testvol-01   | ENABLED | 2097360  | 0      | -      | -      | -      |

5. Manually recreate all file systems and restore the disk group data to the new file systems.

By recreating the low level volume structure using the configuration gathered from the Solaris system, it is possible to save time and accurately maintain the existing physical layout.

**Note:** The procedure outlined above is for use on non-rootdg disk groups only.

### 7.3.3 File system recreation

A file system is a set of files, directories, and other structures that sits in a volume container. The file system itself maintains the information that is required to locate file or directory data on disk. At the lowest level, the file system is comprised of entities like inodes and data blocks. File handling and interaction of the VxFS structures is determined by the file system layout version. For VERITAS Foundation Suite Version 3.4, the file system layout is Version 4. In AIX, for example, JFS and JFS2, have different file system layouts.



Characteristics of a file system are determined at creation by the various options to the **mkfs** command. It is these options that we really want to duplicate in a migration scenario. These details, such as total file system size, block size, inode size, and log size, are known internally to the file system structures, but are not readily accessible. Standard UNIX commands can report information about file systems, and we can gather, somewhat manually, the required details in this way. There are also run-time options associated with a file system, such as those provided at mount time.

On a Solaris system, use:

- ▶ **df -k**: Reports the current file system size and the percentage of the file system that is used/free.
- ▶ **/etc/vfstab** file: Can be used to determine the mount point for a file system and any specified mount options.

If scripts were used to create the original file systems, then the commands within the scripts can be converted for use on AIX. There are some small differences in the command line options between the two platforms; refer to Chapter 6, “Comparisons” on page 181 for more information.

### 7.3.4 Tape backup and recovery

Tape backup and recovery is the traditional method for moving data, and while data volumes remain small, it is a good technique. For very large volumes of data, the times involved in backup and recovery can be prohibitive. Application downtime to take backups must also be taken into account. In some cases, it is possible to reduce the downtime required to make a backup by using the volume snapshot feature of VxVM, which allows a copy of the data to be split off for backup purposes.

We carried out a simple migration test via tape using the VxFS commands **vxdump** and **vxrestore** to back up and restore the contents of a file system. These commands have similar functionality to the **backup** and **restore** commands in AIX and support incremental backups. The **-0** flag will back up the whole file system. We show the migration test in Example 7-6 on page 234. The same procedure that we show here applies also to doing an online backup. See 5.4, “Backups and restores” on page 156 for more information on doing an online backup.

Example 7-6 on page 234 shows commands that can be used to restore the contents of a file system across platforms. This example requires an appropriately sized and named file system to exist on the target platform. By default, **vxrestore** will try, where possible, to maintain the modification time, owner, and mode of the contents of the archive.

```
[On Solaris system]
# vxdump -cf /dev/rmt/0m /test
(remove tape from Sun tape drive)

[On AIX system]
(place tape in AIX tape drive)
# chdev -l rmt0 -a block_size=0
# cd /test
# vxrestore -vx -f /dev/rmt0
(answer '1' for next volume)
(answer 'y' for set owner/mode)
# ls -l /test
# ls -l /test/testdir
```

---

Standard UNIX copy commands, such as **cpio** and **tar**, could also be used to write to and recover files from tape. The suitability of using a particular command must be assessed for the data that is being migrated. Consider the example of a sparse file sometimes used by databases. A sparse file has empty space in it, but real blocks have not yet been allocated. The AIX **backup** and **restore** commands preserve sparseness, but a sparse file backed up with **cpio** would need to be recovered using **pax** to maintain sparseness. The **pax** command would also need to be used to recover a **tar** backup that contained sparse files. If an application has its own backup tool, then this should be used in preference to any other method.

The main advantages of using tape to migrate data are:

- ▶ The data to be migrated is decoupled from the production environment once the backup has been taken.
- ▶ There is potential to use SAN attached tape in conjunction with backup software, such as VERITAS NetBackup or IBM Tivoli Storage Manager, to aid migration of larger amounts of data.

Any migration using tape must take into account that there may be more recent data almost as soon as the backup has been taken. The practical test in Example 7-6 is simple, but demonstrates that backup and recovery via tape is a valid method for migrating files in a VxFS file system from one platform to another.

### 7.3.5 Using network facilities

Every organization has network infrastructure, and this can be used to facilitate migration of data. Data can be copied at the file level or at the raw logical volume level. Network bandwidth and the quantity of data to be migrated will be key

factors in determining how practical migration of data over a network will be for a particular environment.

## Migration of file based data

Let us look at a further example of using **vxdump** and **vxrestore** to back up and recover the contents of a file system, but with the network as the transport mechanism. In this example, based on the setup described in 7.3.1, “Test environment configuration” on page 227, **vxdump** is used to back up the contents of the /test file system to a file. ftp is then used to transport the backup archive to the target server.

On the Solaris system:

1. Back up the contents of the /test file system to a file. Then list the contents to check the archive:

```
# vxdump -f /backup/test_backup /test
# vxrestore -t /backup/test_backup
```

2. Transfer the **vxdump** archive to the target AIX server via ftp:

```
# ftp srvr80e
(login as root)
ftp> bin
ftp> put /aixtest/test_backup
```

On the AIX system:

3. Create a target file system for the data to be migrated, in this case, called /aixtest. Use of the **crfs** command adds an entry to /etc/filesystem:

```
# vxassist make aixtestvol 1g vxdiska01
# crfs -v vxfs -d /dev/vx/dsk/aixdg/aixtestvol -m /aixtest
# mount -v vxfs -o log /aixtest
```

4. Recover the contents of the **vxdump** archive. The -r flag in the **vxrestore** command will recover the whole archive to the current directory. An extra file called restoresymtable is also created. This is used to pass information between incremental restores and can be removed after a -0 level restore:

```
# cd /aixtest
# vxrestore -f /aixtest/test_backup -r
# ls -la /aixtest
....
-rwxr-xr-x  1 user1001 staff      14500 Jul 30 12:48 R50m.jpg
-rwxr-xr-x  1 user1001 staff      4881 Jul 30 12:48 R50s.jpg
-rwxr-xr-x  1 user1001 staff      3072 Jul 30 12:48 S70m.jpg
-rw-r--r--  1 root      system    8840 Jul 31 09:57 restoresymtable
...
# ls -la /aixtest/testdir
```

Working on a file system basis, it is possible to build the **vdump** and **vxrestore** commands into a short pipeline. The commands used to do this are given in Example 7-7. It is assumed in this example that the target file system, for the data to be migrated, already exists as in Step 3 on page 235 for the AIX system.

*Example 7-7 Using rsh with vxdump and vxrestore to migrate files over a network*

---

```
[On Solaris system]
# vxdump 0f - /test | (rsh srvr80e "cd /aixtest; /opt/VRTSvxfs/sbin/vxrestore
rf -")
```

```
[On AIX system]
# ls -la /test
# ls -la /test/testdir
```

---

The use of an archive tool, such as **vxdump**, is beneficial, as it can handle recursive actions, and will back up and recover directory structures as well as content.

**Note:** Use VxFS commands when working with VxFS file systems and AIX commands for JFS or JFS2. **vxrestore** is not supported for recovering files to a non VxFS file system, such as JFS or JFS2. However, the only message received is **vxfs vxrestore: cannot preserve extent attributes**. Even a **sum -r** of the recovered file reported the same value as the originating file, but you should not use **vxrestore** with JFS or JFS2 file systems.

Another common, network based method is to use the Network File System (NFS) to transfer data between networked machines. A file system or directory on one server can be exported to be mounted via the network from a remote server. NFS can also provide a mechanism for transferring data, although its use will place some load on both the client and server systems as well as the network.

### Migration at the volume level

So far, we have examined migration of file based data. It is also possible to migrate raw volumes via the network.

As a first step, clone the volume layout from the Solaris system onto the target AIX system, as outlined in 7.3.2, "Cloning volume layouts" on page 228. Then follow the steps below.

On the Solaris system:

1. Unmount the file system that sits on top of the raw logical volume if there is one:

```
# umount /test
```

On the AIX system:

2. Create a file system if required and leave it unmounted:

```
# crfs -v vxfs -d /dev/vx/dsk/testdg/testvol -m /test
```

On the Solaris system:

3. Initiate a **dd** to copy the contents at the volume level. There is no output from the **dd** to tell you how the copy is progressing. This **dd** took over two hours with a 100 Mbs network connection for a source volume of 1 GB:

```
# dd if=/dev/vx/rdisk/testdg/testvol | (rsh srvr80e dd of=/dev/vx/rdisk/testdg/testvol)
```

The **tcpdump** command can be used to monitor the traffic over an interface and will give an indication that the **dd** copy is progressing:

```
# tcpdump -i en1
```

On the AIX system:

4. Mount the file system, if required, or verify the application can access the raw data. As you remember in 7.3.1, “Test environment configuration” on page 227, we used JPEG files as our data. In this case, the JPEG files were viewed via a Web browser and the execution of the script in /test/testdir confirmed:

```
# mount -v vxfs -o log /test
# ls -la /test
# ls -la /test/testdir
```

Again, the nature of the data and its suitability for cross platform migration should be considered first.

**Note:** A message that contains the text There is a request to a device or address that does not exist., such as in the example below, often simply means the volume has not been started. **vxvol start volname** will resolve this error.

```
# crfs -v vxfs -d /dev/vx/dsk/testdg/testvol -m /test
vxfs mkfs: Cannot open /dev/vx/rdisk/testdg/testvol: There is a request to a
device or address that does not exist.
crfs: 0506-944 Execute module "/sbin/helpers/vxfs/crfs" failed
# vxvol start testvol
```

### 7.3.6 Deport/import of VxVM disk groups

The ideal migration scenario would be to simply connect the existing storage to the new systems and import the required VxVM and VxFS definitions directly from disk. This can be easily achieved in a single platform environment using the deport and import options of the **vxchg** command. Cross platform support for direct import of disk groups is not generally available. It is possible to engage either VERITAS Services or IBM Global Services to migrate external VxVM disk groups from Solaris to AIX.

Conversion support will be provided as part of the professional services offering, in order to enable the cross platform deport and import of data groups that will handle the required header and other changes. This functionality is not available as part of the base VERITAS Foundation Suite for AIX.

Given that data migration across platforms is unlikely to be a common scenario for an organization and will typically happen infrequently, there is a benefit in the current requirement to engage professional services, as this can reduce the level of risk for the data migration portion of a project. There are professionals within both VERITAS and IBM that are focused solely on data migration who have a high level of skill in this area.

The main advantages of importing disk groups as a means of migrating data are:

- ▶ Data is moved, not copied, so no additional physical storage is required, and no additional expenditure is required.
- ▶ The existing storage layout is retained and, assuming the application supports the platform change, data can be available for use almost immediately. However, you must allow for a test phase.
- ▶ Large amounts of data can be migrated in a relatively short time.

The use of external disk subsystems to achieve a greater level of availability and flexibility is a common implementation, and it is fair to say that the importance of cross-platform support for deport and import of disks groups is a known requirement in VERITAS development.

The viability of all of the above relies on the external storage supporting attachment to multiple platforms.

### 7.3.7 Using VERITAS Volume Replicator (VVR)

VERITAS Volume Replicator enables replication of data over IP with full synchronous and asynchronous copy capability. VVR is integrated with the FlashSnap advanced feature of VERITAS Foundation Suite for AIX. In a migration scenario, the ability to support replication of data between hardware

platforms is a key benefit. There is support for replication from Solaris to AIX and vice-versa. Use of VERITAS Volume Replicator requires an additional license.

Setup of VERITAS Volume Replicator is achieved through a Java Graphical User Interface (GUI).

The main benefits of VERITAS Volume Replicator in a migration situation are:

- ▶ The ability to replicate one to many or many to one. So you could replicate data for a production environment and a development environment at the same time.
- ▶ Real-time replication of data is supported, which would allow for parallel running of environments. Fallback to the environment would be relatively straightforward should there be a problem.

Bandwidth and latency of the network must be taken into account for replication over IP.

## 7.4 Migration from AIX LVM to VxVM

As we saw in 6.1.3, “VxVM and LVM co-existence” on page 185, the AIX LVM can co-exist with the VxVM without any problems. So when an application has been implemented with data in AIX volume groups, there are few compelling reasons to move the data across to the VxVM disk group. But, business and management reasons may show the benefits to be gained from implementing VERITAS Foundation Suite for AIX. The existence of AIX volume groups does not prevent the introduction of VERITAS disk groups on the same system and vice-versa.

To coincide with the availability of VERITAS Foundation Suite for AIX, VERITAS have made a tool called **vxvmconvert** available and provided a detailed migration guide. This guide, titled *VERITAS Volume Manager 3.2 - Migration Guide (AIX)*, can be downloaded from <http://support.veritas.com>. Automated migration using **vxvmconvert** and manual conversion are considered next. For completeness, a brief description of the test environment is also included.

### 7.4.1 Test environment configuration

The source and target system in all the examples of migrating LVM and JFS/JFS2 to VxVM and VxFS is a stand-alone RS/6000 F80 running AIX 5L Version 5.1, maintenance level 02, and VERITAS Foundation Suite for AIX. The machines have 32-bit kernels, JFS file systems, and 1 GB of memory each.

## Test configuration 1

In this configuration, the F80 was set up with a single non rootvg volume group called datavg with three mirrored raw logical volumes (each with two copies), as can be seen in Example 7-8.

*Example 7-8 Initial disk configuration for LVM to VxVM conversion*

---

```
[On AIX system]
# lspv
hdisk0          0001615f43bd8180          rootvg
hdisk1          0001615fcbc1a83f          VeritasVolumes
hdisk2          0001615fcbc1a86b          VeritasVolumes
hdisk3          0001615fcbea5d16          datavg
hdisk4          0001615fcbea5e58          datavg
hdisk5          0001615fcbea5f96          datavg
# lsvg -l datavg
datavg:
LV NAME        TYPE      LPs    PPs    PVs    LV STATE    MOUNT POINT
rawlv01        raw       5      15     3      closed/syncd N/A
rawlv02        raw       8      24     3      closed/syncd N/A
rawlv03        raw      10     30     3      closed/syncd N/A
```

---

## Test configuration 2

The F80 was set up to run an http server with the associated documentation held in a file system on a separate volume group. See the command output in Example 7-9 for the initial disk configuration. This scenario was chosen to avoid the time required to install a large application.

*Example 7-9 Initial disk configuration for manual LVM to VxVM migration*

---

```
[On AIX system]
# lspv
hdisk0          0001615f43bd8180          rootvg
hdisk1          0001615fcbc1a83f          VeritasVolumes
hdisk2          0001615fcbc1a86b          VeritasVolumes
hdisk3          0001615fcbea5d16          VeritasVolumes
hdisk4          0001615fcbea5e58          webvg
hdisk5          0001615fcbea5f96          webvg
# lsvg -l webvg
webvg:
LV NAME        TYPE      LPs    PPs    PVs    LV STATE    MOUNT POINT
lv01           jfs       1      2      2      open/syncd
/usr/HTTPServer/htdocs
loglv00        jfslog    1      1      1      open/syncd  N/A
```

---



For completeness, here are the steps required to install and start a simple http documentation server running on AIX 5L Version 5.1:

1. Create a new file system on a non root volume group for the http documentation called /usr/HTTPServer/htdocs. A size of 10 MB should be sufficient for this file system:

```
# crfs
# mount /usr/HTTPServer/htdocs
```

2. Install the following filesets onto the server from the AIX 5L V5.1 Expansion Pack. This will, in addition to installing the http server daemon, populate the /usr/HTTPServer/htdocs directory:

```
# smitty install
(select the following fileset)
- http_server.admin
- http_server.base
- http_server.html
- http_server.man
```

3. Edit httpd.conf in the /usr/HTTPServer/conf and change the ServerName variable to an accessible IP label for this system.
4. Start the http server. To stop the http server, simply use the **apachectl stop** command:

```
# /usr/HTTPServer/bin/apachectl start
```

5. From a Netscape browser, enter the IP label used as the ServerName to access the http documentation server.

## 7.4.2 Converting LVM volume groups to VxVM disk groups

Conversion of LVM volume groups to VxVM disk groups can be achieved using the **vxvmconvert** utility. Conversion of an existing LVM disk to VxVM disk format allows physical reuse of the same piece of storage without any data loss.

The **vxvmconvert** tool converts a volume group in place by changing the names by which the system refers to the storage. Modifications are also required to bring the disks under the control of VxVM, so during this process, there must be no access to data in volume groups undergoing conversion or modification of LVM metadata. Any form of live change has risks associated with it, so testing must be included, and a back out strategy determined at the outset. It is possible to reverse the change of a volume group to a disk group, as long as there have been no changes to the volume structure. The menu for vxmconvert is shown in Example 7-10 on page 242.

#### Example 7-10 vxvmconvert menu options

---

```
# vxvmconvert
Volume Manager Support Operations
Menu: VolumeManager/LVM_Conversion

1      Analyze LVM Volume Groups for Conversion
2      Convert LVM Volume Groups to VxVM
3      Roll back from VxVM to LVM
4      Set path for saving VGRA records
list   List disk information
listvg List LVM Volume Group information

?      Display help about menu
??     Display help about the menuing system
q      Exit from menus
Select an operation to perform:
```

---

The steps to convert an AIX volume group to a VxVM disk group are:

1. Assess the AIX volume group's suitability for conversion and reasons for conversion.
2. Make the appropriate backups and plan a fallback strategy should conversion fail to complete.
3. Stop the applications.
4. Convert a volume group to a VxVM disk group using **vxvmconvert**.
5. Restart the applications.

Of these steps, the most important is the first step. VERITAS provides the **vxvmconvert** tool to automate the conversion process, but the volume groups it can be used on must meet particular criteria, as detailed below.

Specific configurations cannot, at this time, be converted using **vxvmconvert**; they include:

- ▶ The rootvg. This extends to any alternate root volume group that might be installed.
- ▶ Volume groups containing mirrors with the Mirror Write Cache feature turned on.
- ▶ A volume group containing any paging spaces or a /usr file system.
- ▶ Volume groups containing JFS/JFS2 file systems.
- ▶ Volume groups where bad block relocation has occurred. This is related to the fact that VxVM does not carry out bad block relocation at the physical disk level.

In addition, there must be sufficient space in the AIX volume group for VxVM meta data to be overwritten during conversion.

Volume groups with data on raw logical volumes are the best candidates for using **vxvmconvert** to migrate an LVM volume group to a VxVM disk group. For a volume group with file systems, the **vxvmconvert** utility could be used to automate the conversion of the volume layout after manually taking a backup of the file system data.

### Example of converting an LVM volume group

Using the disk setup described in “Test configuration 1” on page 240, data is written to disk. For our data, we used the `/etc/hosts` file, which we copied to each raw logical volume using **dd**. The command used to do this is detailed in Example 7-11.

#### *Example 7-11 Using the dd command to copy data onto a raw logical volume*

---

```
[On AIX system]
# dd if=/etc/hosts of=/dev/rawlv01 bs=32k seek=10
0+1 records in.
0+1 records out.
```

---

In AIX, logical volumes have a logical volume control block (LVCB) that is stored in the first 512 bytes of a logical volume. When using raw logical volumes, not to overwrite the LVCB. The LVCB from `rawlv01` in our test configuration is shown in Example 7-12.

#### *Example 7-12 Output from getlvcb command*

---

```
[On AIX system]
# getlvcb -AT rawlv01
AIX LVCB
intrapolicy = m
copies = 3
interpolicy = m
lvvid = 0001615f00004c00000000ef8c17ca73.1
lvname = rawlv01
label = None
machine id = 1615F4C00
number lps = 5
relocatable = y
strict = y
stripe width = 0
stripe size in exponent = 0
type = raw
upperbound = 32
fs =
```

time created = Thu Aug 8 18:03:48 2002  
time modified = Thu Aug 8 18:03:48 2002

---

To convert datavg:

1. Run **vxvmconvert** from the command line and select option 1 “Analyze LVM Volume Groups for Conversion”. Hit the Enter key to confirm this selection. Informational output is then displayed and is shown in Example 7-13.

*Example 7-13 Informational text from Analyze LVM Volume Groups for conversion*

---

Analyze one or more LVM Volume Groups

Menu: VolumeManager/LVM\_Conversion/Analyze\_LVM\_VGs

Use this operation to analyze one or more LVM Volume Groups for possible conversion to VxVM disk groups. This operation checks for problems that would prevent the conversion from completing successfully. For example, it calculates the space required to add an LVM Volume Group's disk's to a VxVM disk group and to replace any existing LVM partitions and volumes with VxVM Volume Manager volumes, plexes, and sub-disks.

For this release, conversion is only allowed for Non-root LVM Volume Groups. Hence, analysis is only allowed on Non-root LVM Volume Groups.

More than one Volume Group or pattern may be entered at the prompt. Here are some LVM Volume Group selection examples:

listvg: list all LVM Volume Groups  
list: list all disk devices  
vg\_name: a single LVM Volume Group, named vg\_name  
<pattern>: for example: vg04 vg09 vg08

Select Volume Groups to analyze : [<pattern-list>,list,listvg,q,?]

---

Enter the name of the volume group to analyze for conversion, in this case, datavg. **vxvmconvert** will prompt for confirmation of this selection (see Example 7-14).

*Example 7-14 Selection of volume group for pre-conversion analysis*

---

Select Volume Groups to analyze : [<pattern-list>,list,listvg,q,?] datavg

datavg

Analyze this Volume Group? [y,n,q,?] (default: y)

---

Press the Enter key to confirm the selection. The output from the analysis phase prior to conversion of the AIX volume group to a VxVM disk group is captured in Example 7-15.

*Example 7-15 Output from Analyze LVM Volume Groups for conversion*

---

The following disks have been found in the volume group  
and will be analyzed for VxVM conversion.

hdisk3 hdisk4 hdisk5

To allow analysis, a new VxVM disk group, datavg, will be  
fabricated and the disk device hdisk3 will be added to the  
disk group with the disk name datavg01.

The hdisk3 disk has been analyzed and prepared for conversion.

To allow analysis, a new VxVM disk group, datavg, will be  
fabricated and the disk device hdisk4 will be added to the  
disk group with the disk name datavg02.

The hdisk4 disk has been analyzed and prepared for conversion.

To allow analysis, a new VxVM disk group, datavg, will be  
fabricated and the disk device hdisk5 will be added to the  
disk group with the disk name datavg03.

The hdisk5 disk has been analyzed and prepared for conversion.

The first stage of the Analysis process completed successfully.

Second Stage Conversion Analysis of datavg

Analysis of datavg found sufficient Private Space for conversion

Conversion Analysis of the following devices was successful.

hdisk3 hdisk4 hdisk5

Hit RETURN to continue.

---

2. Next, select option 2 “Convert LVM Volume Groups to VxVM”. Hit the Enter key to confirm this selection. Informational text, shown in Example 7-16, is displayed.

*Example 7-16 Informational text from Convert LVM Volume Groups to VxVM*

---

Convert one or more LVM Volume Groups

Menu: VolumeManager/LVM\_Conversion/Convert\_LVM\_VGs

Use this operation to convert LVM Volume Groups to VxVM disk groups. This adds the disks to a disk group and replaces existing LVM volumes with VxVM volumes. LVM-VxVM Volume Group conversion may require a reboot for the changes to take effect. For this release, only Non-root LVM Volume Groups can be converted.

More than one Volume Group or pattern may be entered at the prompt. Here are some LVM Volume Group selection examples:

```
listvg:      list all LVM Volume Groups
list:        list all disk devices
vg_name:     a single LVM Volume Group, named vg_name
<pattern>:   for example:  vg04 vg08 vg09
```

Select Volume Groups to convert : [<pattern-list>,list,listvg,q,?]

---

Enter the name of the volume group to be converted, in this case, `datavg`, and press the Enter key. **vxvmconvert** will prompt for confirmation of this selection (see Example 7-17).

#### *Example 7-17 Selection of volume group for conversion*

Select Volume Groups to convert : [<pattern-list>,list,listvg,q,?] `datavg`

`datavg`

Convert this Volume Group? [y,n,q,?] (default: y)

---

Press the Enter key to confirm the selection. At this point, **vxvmconvert** will prompt the user for an alternative disk group name. Type in desired name and press Enter to confirm (`datadg` was used for this example; see Example 7-18).

#### *Example 7-18 Request for disk group name*

Convert this Volume Group? [y,n,q,?] (default: y)

Specify a name for the new VxVM disk group (default: `datavg`) `datadg`

---

The in place conversion of the volume group to a disk group is initiated by **vxvmconvert**. The output from this step is included in Example 7-19.

#### *Example 7-19 Output from Convert LVM Volume Groups to VxVM*

The following disks have been found in the `datavg` volume group and will be configured for conversion to VxVM disk groups.

`hdisk3 hdisk4 hdisk5`

A new disk group datadg will be created and the disk device hdisk3 will be converted and added to the disk group with the disk name datavg01.

The hdisk3 disk has been analyzed and prepared for conversion.

A new disk group datadg will be created and the disk device hdisk4 will be converted and added to the disk group with the disk name datavg02.

The hdisk4 disk has been analyzed and prepared for conversion.

A new disk group datadg will be created and the disk device hdisk5 will be converted and added to the disk group with the disk name datavg03.

The hdisk5 disk has been analyzed and prepared for conversion.

The first stage of the conversion operation has completed successfully. If you commit to the changes hereafter, the system will attempt to umount all of the associated file systems, stop and export each Volume Group, and then attempt to complete the conversion without having to reboot the system. If we are unable to stop and export any of the Volume Groups, then the conversion process will not be able to complete without a reboot. You would then be given the choice to either abort the conversion, or finish the conversion by rebooting the system.

The conversion process will update the /etc/filesystems file so that volume devices are used to mount the file systems on this disk device. You will need to update any other references such as backup scripts, databases, or manually created swap devices. If you do not like the default names chosen for the corresponding logical volumes, you may change these to whatever you like using vxedit.

Second Stage Conversion Analysis of datavg

Analysis of datavg found sufficient Private Space for conversion

Conversion Analysis of the following devices was successful.

hdisk3 hdisk4 hdisk5

Hit RETURN to continue.

---

Note that the analyze step of **vxvmconvert** will be carried out again during the conversion phase. After hitting the Enter key to continue, **vxvmconvert** prompts for confirmation that the conversion changes should be committed. This request is shown in Example 7-20 on page 248.

*Example 7-20 Request for confirmation to commit conversion changes*

---

Are you ready to commit to these changes? [y,n,q,?] (default: y)

---

Press Enter to confirm conversion of datavg to datadg. The output from this request can be seen in Example 7-21.

*Example 7-21 Output from commit of LVM volume group conversion to VxVM disk group*

---

Saving LVM configuration records for Volume Group datavg ...

LVM Volume Group datavg records saved.

Unmounting datavg file systems..

The Volume Manager is now reconfiguring (initializing phase)...

Volume Manager: Initializing hdisk3 as a converted LVM disk.

Volume Manager: Initializing hdisk4 as a converted LVM disk.

Volume Manager: Initializing hdisk5 as a converted LVM disk.

Volume Manager: Reconfiguration will be done without rebooting.

The Volume Manager is now reconfiguring (initialization phase)...

Volume Manager: Adding datavg01 (hdisk3) as a converted LVM disk.

Volume Manager: Adding datavg02 (hdisk4) as a converted LVM disk.

Volume Manager: Adding datavg03 (hdisk5) as a converted LVM disk.

Adding volumes for hdisk3...

Starting new volumes...

Volume Manager: Converting LVM Volume Groups to VxVM disk groups.

Convert other LVM Volume Groups? [y,n,q,?] (default: n)

---

As there are no further volume groups to convert, select the default option by pressing the Return key to end the “Convert LVM Volume Group to VxVM” phase and return the user to the main **vxdmconvert** menu. Enter q to quit and return to the AIX command line.

3. Run the commands in Example 7-22 on page 249 to view the VxVM configuration that has resulted from the conversion. The **vxdg list** command shows that there is now a disk group called datadg and **vxdisk list** reports the newly converted VxVM disks that comprise this disk group. Finally, the



**vxprint -thg datadg** command shows how the mirrored raw logical volumes have been converted into subdisks, plexes, and volumes.

*Example 7-22 VxVM configuration following conversion of LVM volume group*

```
# vxdg list
NAME          STATE      ID
rootdg        enabled    1027642829.1025.srvr80e
datadg        enabled    1028855780.1149.srvr80e

# vxdisk list
DEVICE        TYPE      DISK      GROUP      STATUS
hdisk0        simple    -         -          LVM
hdisk1        simple    -         -          online
hdisk2        simple    disk01    rootdg     online
hdisk3        simple    datavg01  datadg     online
hdisk4        simple    datavg02  datadg     online
hdisk5        simple    datavg03  datadg     online

# vxprint -thg datadg
DG NAME        NCONFIG      NLOG      MINORS     GROUP-ID
DM NAME        DEVICE       TYPE      PRIVLEN    PUBLEN     STATE
RV NAME        RLINK_CNT    KSTATE    STATE      PRIMARY    DATAVOLS  SRL
RL NAME        RVG          KSTATE    STATE      REM_HOST   REM_DG     REM_RLNK
V NAME         RVG          KSTATE    STATE      LENGTH     READPOL    PREFPLEX
UTYPE
PL NAME        VOLUME      KSTATE    STATE      LENGTH     LAYOUT     NCOL/WID  MODE
SD NAME        PLEX        DISK      DISKOFFS   LENGTH     [COL/]OFF  DEVICE    MODE
SV NAME        PLEX        VOLNAME   NVOLLAYR   LENGTH     [COL/]OFF  AM/NM     MODE
DC NAME        PARENTVOL   LOGVOL
SP NAME        SNAPVOL     DCO

dg datadg      default      default     15000      1028855780.1149.srvr80e

dm datavg01    hdisk3       simple     4096      17769808 -
dm datavg02    hdisk4       simple     4096      17769808 -
dm datavg03    hdisk5       simple     4096      17769808 -

v rawlv01      -            ENABLED    ACTIVE     163840     ROUND      -          gen
pl rawlv01-01  rawlv01      ENABLED    ACTIVE     163840     CONCAT     -          RW
sd datavg03-03 rawlv01-01   datavg03   3571712   163840     0          hdisk5     ENA
pl rawlv01-02  rawlv01      ENABLED    ACTIVE     163840     CONCAT     -          RW
sd datavg02-03 rawlv01-02   datavg02   3571712   163840     0          hdisk4     ENA
pl rawlv01-03  rawlv01      ENABLED    ACTIVE     163840     CONCAT     -          RW
sd datavg01-03 rawlv01-03   datavg01   3571712   163840     0          hdisk3     ENA
pl rawlv01-04  rawlv01      ENABLED    ACTIVE     LOGONLY    CONCAT     -          RW
sd datavg01-04 rawlv01-04   datavg01   0         33         LOG        hdisk3     ENA

v rawlv02      -            ENABLED    ACTIVE     262144     ROUND      -          gen
```

|       |             |            |          |         |         |        |        |     |
|-------|-------------|------------|----------|---------|---------|--------|--------|-----|
| pl    | rawlv02-01  | rawlv02    | ENABLED  | ACTIVE  | 262144  | CONCAT | -      | RW  |
| sd    | datavg03-02 | rawlv02-01 | datavg03 | 3735552 | 262144  | 0      | hdisk5 | ENA |
| pl    | rawlv02-02  | rawlv02    | ENABLED  | ACTIVE  | 262144  | CONCAT | -      | RW  |
| sd    | datavg02-02 | rawlv02-02 | datavg02 | 3735552 | 262144  | 0      | hdisk4 | ENA |
| pl    | rawlv02-03  | rawlv02    | ENABLED  | ACTIVE  | 262144  | CONCAT | -      | RW  |
| sd    | datavg01-02 | rawlv02-03 | datavg01 | 3735552 | 262144  | 0      | hdisk3 | ENA |
| pl    | rawlv02-04  | rawlv02    | ENABLED  | ACTIVE  | LOGONLY | CONCAT | -      | RW  |
| sd    | datavg01-05 | rawlv02-04 | datavg01 | 33      | 33      | LOG    | hdisk3 | ENA |
| <hr/> |             |            |          |         |         |        |        |     |
| v     | rawlv03     | -          | ENABLED  | ACTIVE  | 327680  | ROUND  | -      | gen |
| pl    | rawlv03-01  | rawlv03    | ENABLED  | ACTIVE  | 327680  | CONCAT | -      | RW  |
| sd    | datavg03-01 | rawlv03-01 | datavg03 | 3997696 | 327680  | 0      | hdisk5 | ENA |
| pl    | rawlv03-02  | rawlv03    | ENABLED  | ACTIVE  | 327680  | CONCAT | -      | RW  |
| sd    | datavg02-01 | rawlv03-02 | datavg02 | 3997696 | 327680  | 0      | hdisk4 | ENA |
| pl    | rawlv03-03  | rawlv03    | ENABLED  | ACTIVE  | 327680  | CONCAT | -      | RW  |
| sd    | datavg01-01 | rawlv03-03 | datavg01 | 3997696 | 327680  | 0      | hdisk3 | ENA |
| pl    | rawlv03-04  | rawlv03    | ENABLED  | ACTIVE  | LOGONLY | CONCAT | -      | RW  |
| sd    | datavg01-06 | rawlv03-04 | datavg01 | 66      | 33      | LOG    | hdisk3 | ENA |

- The final step is to confirm that the data on the disk is untouched. Run the command in Example 7-23 and ensure that the contents of /etc/hosts is visible in the data that is retrieved from disk. Note that the raw device name has changed from /dev/rawlv01 to /dev/vx/dsk/datadg/rawlv01.

*Example 7-23 Command to dd data from a raw logical volume*

```
# dd if=/dev/vx/dsk/datadg/rawlv01 of=/tmp/outfile bs=32k skip=>
1+0 records in.
1+0 records out.

# more /tmp/outfile
# @(#)47      1.1  src/bos/usr/sbin/netstart/hosts, cmdnet, bos510 7/24/91
10:
00:46
#
# COMPONENT_NAME: TCPIP hosts
#
# FUNCTIONS: loopback
....
```

**Note:** Any applications or scripts that use device names must be modified to pick up VxVM related names.

While there are no changes to the new VxVM volume configuration, it is possible to reverse the conversion by selecting option 3 “Roll back from VxVM to LVM” in the main **vxvmconvert** menu. The process of rolling back the VxVM configuration was tested in the lab, but the screen output is not documented here. The original

volume group name must be supplied to begin roll back. This process took a little longer than the forward conversion. The roll back also recognized that there were mirrored logical volumes and started a synchronize of the LVM logical volumes as a background task.

### 7.4.3 Manual migration of LVM and JFS to VxVM and VxFS

The **vxvmconvert** tool is unable to handle the conversion of the volume group plus its file systems, so in this section we will step through a completely manual migration from AIX LVM and JFS to VxVM and VxFS. Note that we do not discuss a migration from JFS2 to VxFS. If you are coming from a JFS2 file system, you can do a manual migration, but you will have to ensure that the same features that you are making use of with JFS2 are available with VxFS.

To do a manual migration, the aim is to create the required VxVM objects and VxFS file system to match the existing AIX configuration. For experienced VxVM administrators, determining the required VxVM objects and their options should be relatively straightforward, but for those who are building their skill in this area, there are many options and features to consider, as shown in Figure 7-3 and Figure 7-4 on page 252.

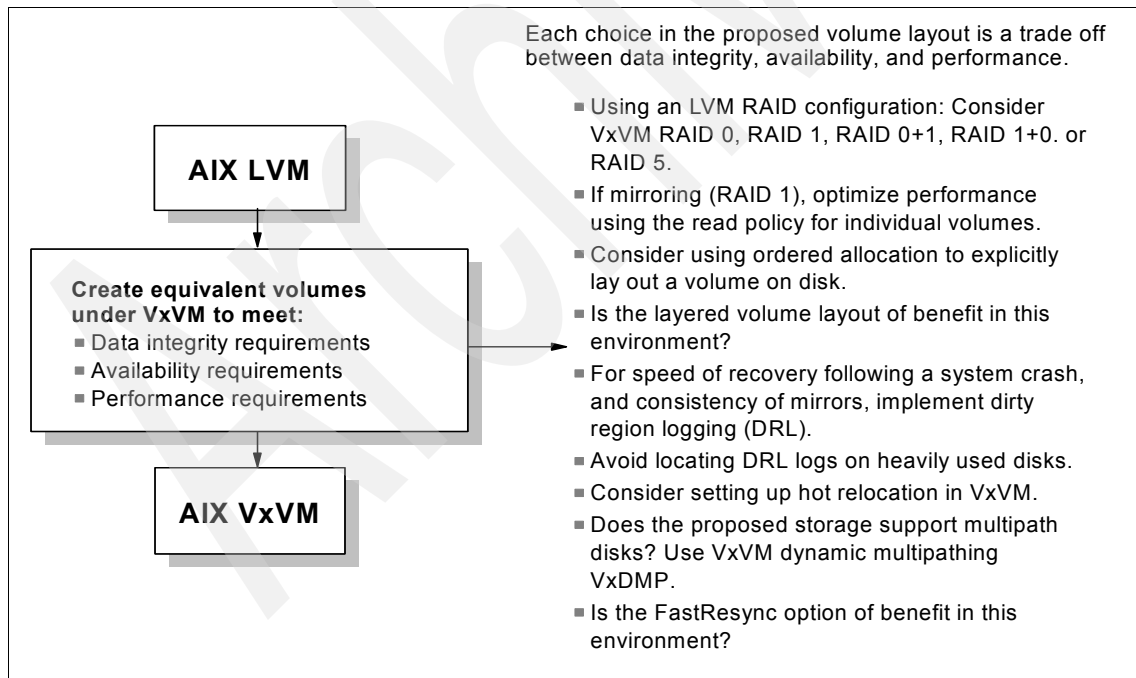


Figure 7-3 VxVM configuration options

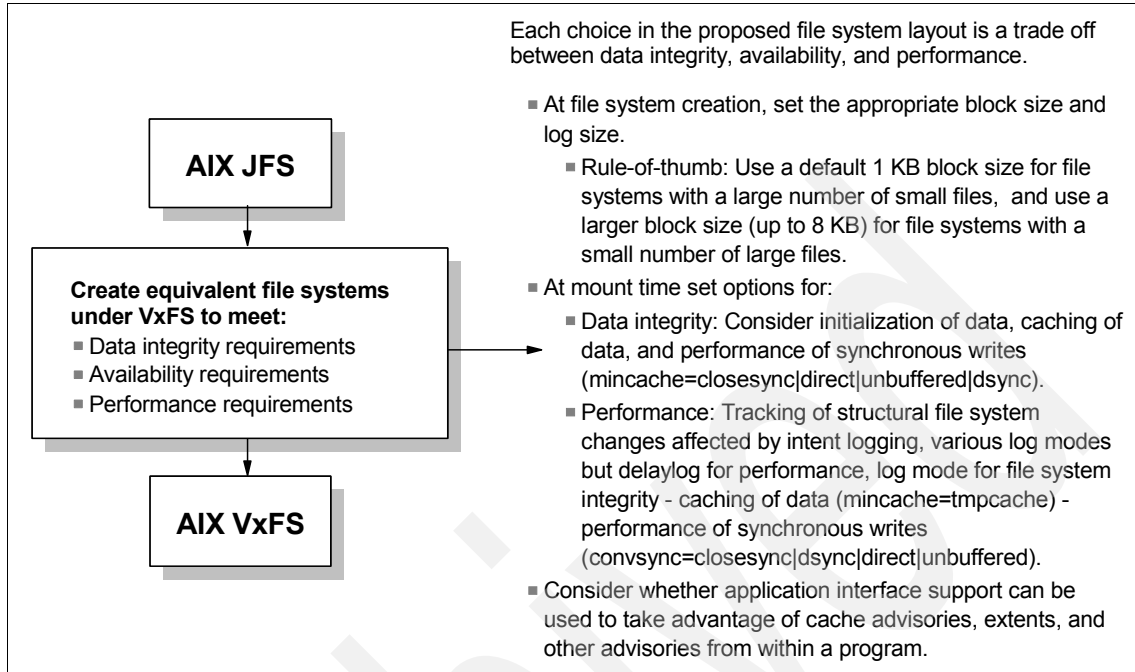


Figure 7-4 VxFS configuration options

### Example of a manual migration of a LVM volume group

Using the setup described in 7.4.1, “Test environment configuration” on page 239, the aim is to migrate a volume group and its associated file systems to a VxVM disk group and VxFS file systems. The volume group we use in our example is webvg. There are a number of practical ways this can be achieved. Here we show one method:

1. Stop the application. To allow migration on a single system, introduce a temporary mount point for all the file systems in the AIX volume group:

```
# /usr/HTTPServer/apachectl stop
# umount /usr/HTTPServer/htdocs
# chfs -m /temp/usr/HTTPServer/htdocs
# mount /temp/usr/HTTPServer/htdocs
```

2. The original configuration is mirrored so data availability is of importance; we need to create a disk group with at least two VM disks in it:

```
# vxdg init vxwebdg disk02=hdisk1
# vxdg -g vxwebdg adddisk disk03=hdisk3
```

```
# vxdg list
NAME          STATE          ID
```

```
rootdg      enabled 1027642829.1025.srvr80e
vxwebdg     enabled 1028305343.1127.srvr80e
```

```
# vxdisk list
DEVICE      TYPE      DISK      GROUP      STATUS
hdisk0      simple    -          -           LVM
hdisk1      simple    disk02     vxwebdg     online
hdisk2      simple    disk01     rootdg      online
hdisk3      simple    disk03     vxwebdg     online
hdisk4      simple    -          -           LVM
hdisk5      simple    -          -           LVM
```

3. Create a mirrored volume to hold a file system for the http server's documentation and also a file system on top of this:

```
# vxassist -g vxwebdg make mirrorvol 1g layout=mirror-concat mirror=2
# vxprint -thg vxwebdg
....
dg vxwebdg      default      default 3000      1028305343.1127.srvr80e
dm disk02       hdisk1       simple 2048      17771856 -
dm disk03       hdisk3       simple 2048      17771728 -
v mirrorvol     -            ENABLED ACTIVE 2097152 SELECT - fsgen
p1 mirrorvol-01 mirrorvol     ENABLED ACTIVE 2097152 CONCAT - RW
sd disk02-01    mirrorvol-01 disk02      0        2097152 0 hdisk1  ENA
p1 mirrorvol-02 mirrorvol     ENABLED ACTIVE 2097152 CONCAT - RW
sd disk03-01    mirrorvol-02 disk03      0        2097152 0 hdisk3  ENA
```

4. Create a file system for the http server documentation and mount it:

```
# crfs -v vxfs -d /dev/vx/dsk/vxwebdg/mirrorvol -g vxwebdg -m \
/usr/HTTPServer/htdocs -a size=20048
version 4 layout
20048 sectors, 10024 blocks of size 1024, log size 1024 blocks
unlimited inodes, largefiles not supported
10024 data blocks, 8928 free data blocks
1 allocation units of 32768 blocks, 32768 data blocks
last allocation unit has 10024 data blocks
```

The log size can be specified using the -a log=# option, or it will be created automatically to a size that depends on the size of the file system:

```
# mount -o log /usr/HTTPServer/htdocs
```

5. Migrate the contents of the file system. A simple copy will work for this small example. Care must be taken with any direct copy commands that the command being executed will have the intended effect. It is easy to make a mistake and overwrite good data:

```
# cd /temp/usr/HTTPServer/htdocs
# cp -R * /usr/HTTPServer/htdocs
```

Even in this test configuration, doing a backup and restore in two steps would be cleaner and less prone to error. However, space would be required for the backup.

6. Start the http server and confirm that it is possible to view the documentation via a Web browser:

```
# /usr/HTTPServer/bin/apachectl start
```

7. Unmount the source file system and remove the volume group definition from the AIX system. This will leave the source physical volumes available once more for use by the LVM or the VxVM. The **rmfs** command should be run before the volume group definition is exported to remove the `/etc/filesystem` entry and mount point, if desired:

```
# umount /temp/HTTPServer/htdocs
# rmfs
# exportvg webvg
# lspv
```

**Note:** The **tar** and **cpio** commands cannot be used to back up and restore files larger than 2 GB, due to adherence to POSIX standards.

It seems likely that the most common scenario for implementing VxVM disk groups will be where new servers and, potentially, storage have been purchased for new projects. In this case, data migration is not typically required, as new data will be created when the systems go into production, so the more limited support that exists for migrating AIX LVM to VxVM and JFS/JFS2 to VxFS should have a minimal impact. Plan to provide a **vxfsconvert** command to handle JFS/JFS2 file system conversion. Please see 9.4, “Administration issues” on page 295 for more information.

## 7.5 Other migration scenarios

For completeness here are some brief comments on migration of data between servers of the same type running VERITAS Foundation Suite and also migration of a UNIX file system to VxFS on AIX. These two migration scenarios are included here but are not explicitly outlined in Figure 7-2 on page 223.

### 7.5.1 Single platform migration

Standard import and deport of disk groups between systems of the same type facilitates ease of movement of data and is the first step to high availability. The use of external disks decouples the data from the server. This can reduce downtime if there is a complete server failure, as the external disks containing the data can simply be attached to another system and the definitions imported,

allowing access to the data. Import and deport of disk group definitions within a single platform is fully supported by VxVM and is commonly used in much the same way as exportvg and importvg are used in an AIX LVM environment on an IBM @server pSeries server.

The caveat for deport and import of disk groups that contain file systems is that the target system must support the file system layout implemented on the source system. Any version of VxFS that supports a later file system layout will typically handle the import of a disk group with an earlier layout, but the reverse is not true. Note that VERITAS Foundation Suite for AIX Version 3.4 only supports the VERITAS Version 4 file system layout. This way of implementing support for backwards compatibility is standard practice and, as long as it is understood, can be accounted for in a production environment.

**Note:** VERITAS Foundation Suite for AIX Version 3.4 only supports the Version 4 file system layout.

In a single platform migration, there should, at the software level, be little or no conversion or porting actions required. The release notes for the intended software will highlight any changes required. If the move does involve the introduction of a new release of any component of the software solution, then comprehensive testing should be carried out to prove the production environment.

### 7.5.2 Migration of a UNIX File system (UFS) to VxFS on AIX

For ease of administration and availability reasons, such as improved recovery following an unexpected outage, it is beneficial to bring UFS file systems under the control of a volume manager and production quality file system. Volume and file system layout will need to be designed and implemented for the target system prior to migration of data. There are no practical examples included in this chapter of migration from UFS to VxFS on AIX. However, the same steps currently required to migrate AIX JFS and JFS2 to VxFS apply here.

Specific choices, some of which are outlined in Figure 7-3 on page 251, need to be planned for to take advantage of the additional functionality offered by VxFS over the standard UFS. Some time may need to be taken, and testing carried out, to determine the real effects of particular VxFS or VxVM configuration with the existing application.

## 7.6 Summary and recommendations

This chapter has shown, via a number of practical examples, that migration can be accomplished. From the tests that have been carried out for this redbook, it is clear that any migration effort must be planned and the suitability of the application(s) on the intended platform be confirmed as the first requirement.

The least risk and resource-intensive option to migrate VERITAS Foundation Suite on Solaris to AIX is to physically attach the existing storage to the AIX system and to engage professional services for direct import and conversion of disk groups. If the existing storage cannot be attached to the AIX server, then depending on the skills in house, it would probably still be worth employing the services of specialists who are skilled in data migration to assess and plan for a tape or network oriented migration.

Support for cross platform migration to VERITAS Foundation Suite for AIX is a known requirement and is recognized as a critical area for VERITAS. It is expected that there will be continued development in this area. However, remember that VERITAS Foundation Suite for AIX is still only one component of the whole picture migration picture.

The individual sections in this chapter have highlighted the pros and cons of the various migration paths described in Figure 7-2 on page 223, but it must be recognized that any migration will require time and resources to plan and implement successfully.



# Performance, tuning, and scalability

This chapter discusses introductory aspects of performance and tuning in relation to VERITAS Foundation Suite for AIX. VxVM and VxFS both have default configuration options that are suitable for the majority of cases. Various performance features/enhancements in VxVM and VxFS are discussed in this chapter, and scalability is also commented on.

Specific topics covered include:

- ▶ Basic performance guidelines for VxVM and VxFS
- ▶ Monitoring functions in VxVM and VxFS
- ▶ Tuning parameters

## 8.1 Basic performance guidelines for VxVM and VxFS

There are basic performance guidelines that apply to the management of any storage subsystem, volume manager, and file system. Performance goals can differ depending on the business requirements, for example, throughput might be important in one situation, and application response time in another. The best results are achieved when the behavior of the primary application workloads are understood, the ratio of reads to writes, and whether the I/O is random or sequential. Other requirements must also be taken into account, such as data integrity and availability. This section discusses some of the configuration options for VxVM and VxFS that can influence performance.

### 8.1.1 Physical disks and data assignment

Performance strategies for VxVM focus on balancing I/O load, and using striping and mirroring to maximize I/O bandwidth to frequently used volumes. However, physical disk configuration and layout of data can also impact performance and should be also be considered. Layout of data is to some extent the responsibility of the system administrator and, as discussed in 7.2.1, “Applications” on page 223, some application vendors publish best practices documents. There are a number of basic guidelines that should be followed for any application setup.

At the physical disk level, placement of disk adapters, the number of disk adapters, the type of disks, and the cabling of the disk subsystem must be considered carefully for performance. For example, the system may have a number of fast adapters, but the bus can become a performance bottleneck if it is unable to support them. The appropriate adapter placement guide for any given UNIX server should be consulted to ensure that the bus supports the intended adapters. For IBM @server pSeries systems, the *PCI Adapter Placement Guide* can be downloaded from:

[http://www.ibm.com/servers/eserver/pseries/library/hardware\\_docs/](http://www.ibm.com/servers/eserver/pseries/library/hardware_docs/)

The VERITAS Volume Manager supports disk groups comprised of disks of different capacity and types. However, mixing disks with varying performance specifications in a volume can lead to odd results when measuring performance, especially when using a striped or mirrored configuration. The recommendation is, therefore, to use similar disks throughout a volume where possible.

With VxVM, it is possible to spread a logical volume over multiple physical volumes to take advantage of the underlying physical configuration, for example, disk access via multiple controllers. In general, frequently accessed (hot) logical volumes should be spread over multiple physical volumes. For an application, the slowest operation is typically getting data to and from disk. Read and write

operations imply disk head movements, so storing data contiguously on the underlying physical disk is good start to minimize disk head operations. Specific allocation requirements can be met when creating subdisks. Ordered allocation provides a more flexible approach to this process and is described in “Allocation policy” on page 201.

For mirroring, the mix of read to writes will determine whether mirroring can offer any real performance gain. Typically, at least 70% of the physical I/Os need to be reads to see a performance benefit over an unmirrored environment. Obviously, the availability benefits of mirroring are strong. Mirroring across disks or luns attached to different controllers can increase performance, and if a particular plex of a mirror offers better performance than another, the read policy can be set to favor the faster plex.

For striping, 64 KB is a good stripe size to start with if the application's I/O pattern is unknown or it is not possible to set the stripe unit size to the track size. This is not the default stripe size. Spreading stripes over a greater number of physical disks can improve performance but, at the same time, reduces the effective mean time between failure of the volume. However, the striped volume could then be mirrored to improve availability. If possible, distribute subdisks in a striped volume across different controllers and buses. Care must be taken when configuring volumes to not place heavily used unstriped volumes on a disk that contains a striped volume. Otherwise, the performance of the striped volume can be adversely affected.

Coping with future requirements is one difficulty when planning data assignment. Once data access patterns are established, VxVM supports re-arranging volume layouts online without interrupting availability, and with no requirement to back up and recover data. Careful data assignment using VxVM volume layouts, coupled with the underlying physical disk setup, enables reading and writing data to be done efficiently.

### 8.1.2 VxVM logs

There are a number of optional logs associated with VxVM volumes that are used for data integrity and to speed up synchronization of mirrors. These include the Dirty Region Log (DRL), and the Data Change Object (DCO) log for persistent FastResync. RAID 5 VxVM volumes can also have a RAID 5 log associated with them. With RAID 5, it is understood that there will be a performance impact associated with its use, but recovery with a RAID 5 log will be significantly faster, as updates only need to be made to the data and parity portions of the volume that were in transit during a failure. So the entire volume does not have to be resynchronized.

# Performance and the DRL

The DRL is essentially a bit map used to represent a volume, where the bit in the DRL is marked dirty before a write to the associated region of the volume is executed. Writes to the DRL are synchronous, and although they do not occur for every write, it is beneficial to keep the DRL off heavily used disks. There will be a performance hit associated with dirty region logging as additional writes to the DRL are required for real writes to disk.

DRL provides transactional integrity at the volume level and is required to obtain consistency between mirrors. The cost of resilvering (resynchronizing) an entire mirror after a crash can sometimes be prohibitive and many organizations accept the performance loss from maintaining the DRL for the availability offered. It is possible to alter the size of the dirty region log, and therefore the number and size of regions that it is capable of tracking.

Included next is a simple example, which compares recovery times for volumes with and without DRL logging enabled. Example 8-1 includes the `vxprint` output for the mirrored volume without, and Example 8-2 shows a similar volume, with the same size and number of plex copies, but with a single DRL log enabled. The log subdisk is small (33 sectors of 512 bytes). Transactional integrity must also be considered at the file system level, and intent logging is discussed in 8.1.6, “Mount command options” on page 266. This test concentrates on what is happening at the volume level.

Example 8-1 Volume setup without DRL enabled

|                          |           |           |        |                 |         |         |       |    |  |
|--------------------------|-----------|-----------|--------|-----------------|---------|---------|-------|----|--|
| # vxprint -thg veritasdg |           |           |        |                 |         |         |       |    |  |
| ...                      |           |           |        |                 |         |         |       |    |  |
| dg veritasdg             | default   | default   | 10000  | 1029528840.1197 | srvr80e |         |       |    |  |
| dm veritas01             | hdisk11   | simple    | 2048   | 4401929         | -       |         |       |    |  |
| dm veritas02             | hdisk12   | simple    | 2048   | 4401929         | -       |         |       |    |  |
| dm veritas03             | hdisk13   | simple    | 2048   | 4401929         | -       |         |       |    |  |
| v mirvol                 | -         | ENABLED   | ACTIVE | 200000          | SELECT  | -       | fsген |    |  |
| pl mirvol-01             | mirvol    | ENABLED   | ACTIVE | 200000          | CONCAT  | -       |       | RW |  |
| sd veritas01-01          | mirvol-01 | veritas01 | 0      | 200000          | 0       | hdisk11 | ENA   |    |  |
| pl mirvol-02             | mirvol    | ENABLED   | ACTIVE | 200000          | CONCAT  | -       |       | RW |  |
| sd veritas02-01          | mirvol-02 | veritas02 | 0      | 200000          | 0       | hdisk12 | ENA   |    |  |
| pl mirvol-03             | mirvol    | ENABLED   | ACTIVE | 200000          | CONCAT  | -       |       | RW |  |
| sd veritas03-01          | mirvol-03 | veritas03 | 0      | 200000          | 0       | hdisk13 | ENA   |    |  |

Example 8-2 Volume layout with DRL enabled

|                       |         |         |       |                 |         |  |  |  |  |
|-----------------------|---------|---------|-------|-----------------|---------|--|--|--|--|
| # vxprint -thg testdg |         |         |       |                 |         |  |  |  |  |
| ...                   |         |         |       |                 |         |  |  |  |  |
| dg testdg             | default | default | 19000 | 1029957455.1211 | srvr80e |  |  |  |  |

|                  |               |          |        |         |        |         |     |
|------------------|---------------|----------|--------|---------|--------|---------|-----|
| dm testdg01      | hdisk8        | simple   | 2048   | 4401929 | -      |         |     |
| dm testdg02      | hdisk9        | simple   | 2048   | 4401929 | -      |         |     |
| dm testdg03      | hdisk10       | simple   | 2048   | 4401929 | -      |         |     |
| <hr/>            |               |          |        |         |        |         |     |
| v testmirvol     | -             | ENABLED  | ACTIVE | 200000  | SELECT | - fsgen |     |
| pl testmirvol-01 | testmirvol    | ENABLED  | ACTIVE | 200000  | CONCAT | -       | RW  |
| sd testdg01-01   | testmirvol-01 | testdg01 | 33     | 200000  | 0      | hdisk8  | ENA |
| pl testmirvol-02 | testmirvol    | ENABLED  | ACTIVE | 200000  | CONCAT | -       | RW  |
| sd testdg02-01   | testmirvol-02 | testdg02 | 0      | 200000  | 0      | hdisk9  | ENA |
| pl testmirvol-03 | testmirvol    | ENABLED  | ACTIVE | 200000  | CONCAT | -       | RW  |
| sd testdg03-01   | testmirvol-03 | testdg03 | 0      | 200000  | 0      | hdisk10 | ENA |
| pl testmirvol-04 | testmirvol    | ENABLED  | ACTIVE | LOGONLY | CONCAT | -       | RW  |
| sd testdg02-02   | testmirvol-04 | testdg02 | 200000 | 33      | LOG    | hdisk9  | ENA |

---

This example demonstrates that where a plex has become stale, DRL has no effect on recovery time, so the entire mirror has to be resynchronized. However, where there has been a system crash, the use of a DRL log can significantly speed up the recovery process.

To demonstrate how DRL affects recovery time, let us first simply detach and reattach a plex (or mirror) from the volumes shown in Example 8-1 on page 260 and Example 8-2 on page 260. The **vxprint** output for both volumes shows that the plex is in the STALE state once it is detached:

```
# vxplex det mirvol-01
# vxprint -thg veritasdg
...
v mirvol - ENABLED ACTIVE 200000 SELECT - fsgen
pl mirvol-01 mirvol DETACHED STALE 200000 CONCAT - RW
# vxplex det testmirvol-01
# vxprint -thg testdg
...
v testmirvol - ENABLED ACTIVE 200000 SELECT - fsgen
pl testmirvol-01 testmirvol DETACHED STALE 200000 CONCAT - RW
```

Recovery with the **vxrecover** command is timed to determine how long it took to sync to the stale mirror. The **vxrecover** command performs a plex attach and resync, and can be run in the foreground or background. The results for the volume *without* DRL are shown first, followed by those for the volume *with* DRL enabled:

```
# time vxrecover mirvol
real 0m27.46s
user 0m0.02s
sys 0m0.20s

# time vxrecover testmirvol
```

```
real    0m27.49s
user    0m0.01s
sys     0m0.20s
```

The next step simulates a system crash. Intensive write activity is initiated by issuing the **dd** command, and then a forced stop of each of the volumes is carried out:

```
# dd if=/dev/zero of=/dev/vx/rdisk/veritasdg/mirvol bs=1k count=10000 &
# vxvol -g veritasdg -f stop mirvol
# vxrpint -th mirvol
...
v mirvol - DISABLED NEEDSYN 200000 SELECT - fsgen
pl mirvol-01 mirvol DISABLED ACTIVE 200000 CONCAT - RW
sd veritas01-01 mirvol-01 veritas01 0 200000 0 hdisk11 ENA
pl mirvol-02 mirvol DISABLED ACTIVE 200000 CONCAT - RW
sd veritas02-01 mirvol-02 veritas02 0 200000 0 hdisk12 ENA
pl mirvol-03 mirvol DISABLED ACTIVE 200000 CONCAT - RW
sd veritas03-01 mirvol-03 veritas03 0 200000 0 hdisk13 ENA

# dd if=/dev/zero of=/dev/vx/rdisk/testdg/testmirvol b2=1k count=10000 &
# vxvol -g testdg -f stop testmirvol
# vxrpint -th testmirvol
...
v testmirvol - DISABLED NEEDSYN 200000 SELECT - fsgen
pl testmirvol-01 testmirvol DISABLED ACTIVE 200000 CONCAT - RW
sd testdg01-01 testmirvol-01 testdg01 33 200000 0 hdisk8 ENA
pl testmirvol-02 testmirvol DISABLED ACTIVE 200000 CONCAT - RW
sd testdg02-01 testmirvol-02 testdg02 0 200000 0 hdisk9 ENA
pl testmirvol-03 testmirvol DISABLED ACTIVE 200000 CONCAT - RW
sd testdg03-01 testmirvol-03 testdg03 0 200000 0 hdisk10 ENA
pl testmirvol-04 testmirvol DISABLED ACTIVE LOGONLY CONCAT - RW
sd testdg02-02 testmirvol-04 testdg02 200000 33 LOG hdisk9 ENA
```

Both volumes are now in the **NEEDSYN** state. Again, the **vxrecover** command is used, but this time with the **-s** option to restart disabled volumes. The results for the volume *without* the DRL are shown first, followed by those for the volume *with* DRL enabled:

```
# time vxrecover -s mirvol
real    0m28.00s
user    0m0.00s
sys     0m0.25s

# time vxrecover -s testmirvol
real    0m1.00s
user    0m0.01s
sys     0m0.07s
```

In this case, there is a significant difference in the time required to recover the volume, and it is clear that the DRL is beneficial in this scenario.

## Performance and FastResync

FastResync keeps track of writes to unavailable mirrors, and works by copying only changes to the newly reattached volume based on a data change map held in memory or on disk. This process reduces the time required to rejoin a split mirror, and requires less processing power than a full mirror resynchronization without logging, should a plex become stale. With persistent FastResync, the Data Change Object (DCO) log stores the data change map on disk, and records updates that have been missed by the failed or detached mirror. The location of DCO logs can impact performance. In general, a DCO log should not be located on the same physical disks as the application data, especially where the volumes are write intensive.

### 8.1.3 Extent-based allocation

Traditional UNIX file systems typically employ block-based allocation, simply allocating blocks linearly off of a free list, or adding the minor optimization of an interleave factor. Block allocating file systems are subject to significant fragmentation, reducing overall performance. As described in 2.3.3, “Extent-based allocation” on page 36, VxFS uses extent-based allocation instead of block-based allocation. With extent-based allocation, it is possible to define a contiguous range of blocks for inclusion in a file, based on either the I/O pattern of the application, or explicit requests by the user or programmer. This approach allows larger I/O operations, which are more efficient, to be passed to the underlying drivers, increasing performance for most applications. In addition, because a single index pointer can address many blocks of a file, fewer indirect pointers are required in the meta data, making access far more efficient for medium-sized and large files.

Extent attributes are associated with a file and the **getext** and **setext** commands can be used to view or edit extent attributes. A file must exist to be able to use **setext**. An example of how to view the extent attributes is shown in Example 8-3 on page 265. The two primary extent attributes are related to space reservation policies, and using a non-default fixed extent. Reservation attributes allow disk space to be pre-allocated, and to define what happens to allocated but unused space when a file is closed. Critical logs can have space pre-allocated so that a full file system will not affect their use. The use of fixed length extents can reduce the number of allocations required by an application. Applications can use the **VX\_SETTEXT** ioctl. For more detailed information about the flags associated with **VX\_SETTEXT**, see “Extent Information” in Chapter 4, “Application Interface”, in the *VERITAS File System 3.4 - Administrator's Guide 3.4 (AIX)*.

Extents are also used by inodes. Section 2.3.4, “Inodes” on page 37 describes how VxFS inodes reference a defined number of direct and indirect extents. The **vxtune fs** command detailed in 8.3.4, “File system tuning parameters” on page 273 can be used to change the default size of the indirect extent.

The decision whether to interact and modify the default extent-based allocation policies will depend on the application and the file type. Under most circumstances, the default policies will satisfy performance requirements.

## 8.1.4 Inode and directory optimizations

Inodes and directories are very frequently accessed. UNIX operating systems attempt to optimize their access using caching techniques. VxFS has several optimizations that may be used to speed access of these commonly used structures.

### Hashed directories

The VERITAS file system supports variable-length directory entries. Because file names may be very long, rather than allocate every directory entry at maximum size, wasting directory space and slowing access time, variable-length entries are used. However, these are less efficient to search than fixed-sized entries, because it is not known where the next entry begins. To eliminate the need for a linear directory search, VxFS uses a hash table in each directory extent to locate directory entries quickly, accelerating file access.

### Immediate objects

The VERITAS file system attempts to accelerate access to commonly used “files” that are used exclusively to look up other files. Symbolic links and directories that are small enough to fit in a reserved “immediate area” in the inode (currently 96 bytes) are stored there, rather than in a separate data extent. This saves access time and storage. The directory pointer optimization listed above makes it more likely that directories will be small enough to be stored as immediate objects.

## 8.1.5 VxFS create options

There are two file system parameters that can be set at creation time: block size and intent log size.

### Block size

The block size for a file system is the minimum amount of space allocated to a file. By default, this logical block size is 1 KB. The block size at the disk level is 512 bytes, as detailed in “Physical disks” on page 194. The **mkfs** command with the **-o bsize** option is used to define the block size for a file system, in order to



take advantage of the extent-based allocation policy. See 2.3.3, “Extent-based allocation” on page 36 for details of default block sizes used with various file system sizes. The default values will work best in most cases. Larger block sizes use less disk space in terms of file system overhead but more in terms of file allocation. One rule of thumb is to use the default 1 KB block size for a file system with a large number of small files and a larger block size (up to 8 KB) for applications (file systems) with only a few large files.

**Note:** The file system block size is set when a file system is created and cannot be changed.

## Intent log size

The intent log is used for file system journaling in VxFS and keeps track of changes to file system structure. Section 2.3.6, “Journaling” on page 38 describes how the intent log is used during file system recovery. By default, the intent log size is 1024 blocks. There are some scenarios where a larger intent log will improve performance, such as those with intensive synchronous writes. However, file system recovery time is dependent on log size. So the larger the log, the longer the time required to recover a file system. The **mkfs** command with the **-a logsize** option is used to change the default size of the intent log.

**Note:** The intent log size is set when a file system is created and cannot be changed.

Example 8-3 shows how the block size and intent log size can be set at file system creation. The file system was formatted on the mirrored volume, **mirvol**, shown in the **vxprint** command output in Example 8-1 on page 260. The volume was created via the **smitty vxvmlvadd** fast path; select “Add a Mirrored VxVM Volume” with the default options accepted and the file system created via **smitty crvxfslvstd**. The default block size of 1 KB is okay for this file system, which is less than 8 GB. The intent log size was increased from the default to 1096 blocks. An extract from the **/smit.log** is shown for this action. A large file was then written to the file system, and the default output from the **gettext** command is included for completeness.

*Example 8-3 Creation of a file system - set block size and intent log size*

```
# more /smit.log
...
Command_to_Execute follows below:
>> /sbin/helpers/vxfs/crfs -v vxfs -d'mirvol' -g'veritasdg' '-Tvv'-m'/mirrorfs'
-A''\locale nostr | awk -F: '{print $1}' -a version='4' -a logsize='1096'
```

Output from Command\_to\_Execute follows below:

```

---- start ----
  version 4 layout
  200000 sectors, 100000 blocks of size 1024, log size 1096 blocks
  unlimited inodes, largefiles not supported
  100000 data blocks, 98808 free data blocks
  4 allocation units of 32768 blocks, 32768 data blocks
  last allocation unit has 1696 data blocks
---- end ----

# /opt/VRTSvxfs/sbin/gettext -V vxfs -f tempfile_4MB
Bsize 1024  Reserve      0  Extent Size      0

```

---

## 8.1.6 Mount command options

VxFS offers additional options to the standard mount commands, which can provide significant performance improvements. This section discusses the most common methods for enhancing file system performance, the benefits, and potential drawbacks.

### Intent logging modes

Control of the intent logging mode of a file system at mount time is a primary method for increasing performance. The type of workload and associated I/O operations will influence the level of performance improvement. File system structure intensive loads may show a significant improvement, with *delaylog* and *tmplog*, but read/write intensive loads might only show a negligible improvement.

Intent logging or journaling is essential for rapid file system recovery should there be a system crash. File system logging provides integrity of file system transactions, which are things like deleting a file or updating an inode. Intent logging ensures that the file system is always consistent, or, more accurately, that it can always be returned to the last consistent state without requiring a full scan (**fsck**).

Intent logging modes are set at mount time with the **mount -V vxfs -o *intent\_mode***. The default mode used at mount time is the delaylog; however, it is only the log mode that guarantees *all* structural changes are logged on disk when the system call returns. The best way to select a mode is to test a workload representative of the production environment against different logging modes and to compare the performance. A full list of intent logging modes is included in 4.3.6, “Mounting file systems” on page 122. See “Performance of log and delaylog mode intent logging” on page 267 for an example of delaylog mode performance compared with log mode.

### ***The delaylog mode***

With delaylog mode, not all entries in the intent log are written before the system call is returned. This logging delay improves the performance of the system, but some changes are not guaranteed until a short time after the system call returns, when the intent log is written. If a system failure occurs, recent changes may be lost. This mode approximates traditional UNIX guarantees for file system correctness in case of system failures. Fast file system recovery (replaying the intent log) works with this mode.

**Note:** The delaylog mode is the default mode at file system mount.

### ***The tmplog mode***

In tmplog mode, intent logging is nearly always delayed. This greatly improves performance, but recent changes may be lost if the system crashes. Fast file system recovery also works with this mode.

**Note:** The tmplog mode is *only* recommended for temporary file systems.

## **Performance of log and delaylog mode intent logging**

This test is based on the file system described in Example 8-3 on page 265. Using the default intent logging mode for **mount**, the script in 8.2.1, “vxstat and vxtrace” on page 268 was run to create 2000 empty files in the file system. This process was timed and the results are shown here:

```
# mount
...
/dev/vx/dsk/veritasdg/mirvol /mirrors vxfs Aug 19 15:38
rw,delaylog,suid,ioerror=mwdisable,nolargefiles
# time /tmp/mini
real    0m12.99s
user    0m1.02s
sys     0m12.07s
```

The temporary files in the /mirrors file system were removed, the file system unmounted, then mounted again, using the log intent mode, and the script to create files was re-run. Again, the file creation process was timed:

```
# cd /mirrors
# rm file*
# umount /mirrors
# mount -V vxfs -o log /mirrors
# mount
...
/dev/vx/dsk/veritasdg/mirvol /mirrors vxfs Aug 20 10:28
rw,log,suid,ioerror=mwdisable,nolargefiles
# cd /mirrors
```

```
# time /tmp/mini
real    0m50.98s
user    0m1.05s
sys     0m12.55s
```

In this example, delaylog significantly outperformed the log, taking only 12.99 seconds, compared with 50.98 seconds.

## Caching options

Caching behavior can also be altered at mount time to improve performance.

The **mount -o mincache=tmpcache|closesync|direct|dsync|unbuffered** alters the caching behavior of a file system. Specifically, the tmpcache option flushes the file at times other than when the file is closed. This is a direct trade off between integrity and performance.

Similarly, the **mount -o convosync=direct|dsync|unbuffered|closesync|delay** command can be used to optimize performance of synchronous writes, altering the caching behavior of the file system for O\_SYNC and O\_DSYNC I/O operations.

## 8.2 Monitoring VxVM and VxFS

Tools for monitoring VxVM and VxFS are provided as part of the VERITAS Foundation Suite for AIX.

### 8.2.1 vxstat and vxtrace

VxVM provides the **vxstat** and **vxtrace** commands to gather performance data. On an existing system, the **vxstat** command, for example, can be used to get the average read and write access times per volume, and provides data that allows you to calculate the average read and write sizes in blocks. The **vxtrace** command traces all I/O for a given volume, giving the location, size, and time to complete for each individual I/O.

The data from **vxstat** is cumulative from reboot or the last time the statistics were reset. Statistics are read from the volume device files in the /dev/vx/rdisk directory. In a similar way to the AIX **iostat** command, **vxstat** output can be captured at intervals using the -i interval flag, or as a snapshot in time. Example 8-4 on page 269 shows the **vxstat** disk group, subdisk, and plex output for a mirrored volume, after the following simple script was run to create some files:

```
# vi /tmp/mini
```

```

!/bin/ksh
i=0
while (( i <= 2000 ))
do
    touch file$i
    (( i = i + 1 ))
done

```

A count of I/O operations is included in the output, and the number of blocks transferred. Multiple blocks may be transferred in a single operation, as can be seen in our example. The average (AVG) time is related to operation of the VxVM interface and should not be compared with statistics from other programs. Statistics are gathered for all logical I/Os.

*Example 8-4 vxstat output for file system with delaylog mode for intent logging*

---

```
# vxstat -g veritasdg
```

|     |        | OPERATIONS |       | BLOCKS |       | AVG TIME(ms) |       |
|-----|--------|------------|-------|--------|-------|--------------|-------|
| TYP | NAME   | READ       | WRITE | READ   | WRITE | READ         | WRITE |
| vol | mirvol | 0          | 143   | 0      | 6161  | 0.0          | 22.2  |

```
# vxstat -s mirvol
```

|     |              | OPERATIONS |       | BLOCKS |       | AVG TIME(ms) |       |
|-----|--------------|------------|-------|--------|-------|--------------|-------|
| TYP | NAME         | READ       | WRITE | READ   | WRITE | READ         | WRITE |
| sd  | veritas01-01 | 0          | 144   | 0      | 6164  | 0.0          | 20.8  |
| sd  | veritas02-01 | 0          | 144   | 0      | 6164  | 0.0          | 20.6  |
| sd  | veritas03-01 | 0          | 144   | 0      | 6164  | 0.0          | 19.6  |

```
# vxstat -p mirvol
```

|     |           | OPERATIONS |       | BLOCKS |       | AVG TIME(ms) |       |
|-----|-----------|------------|-------|--------|-------|--------------|-------|
| TYP | NAME      | READ       | WRITE | READ   | WRITE | READ         | WRITE |
| pl  | mirvol-01 | 0          | 144   | 0      | 6164  | 0.0          | 20.8  |
| pl  | mirvol-02 | 0          | 144   | 0      | 6164  | 0.0          | 20.6  |
| pl  | mirvol-03 | 0          | 144   | 0      | 6164  | 0.0          | 19.6  |

---

The **-r** option of the **vxstat** command is used to reset volume statistics. It is possible to reset statistics for the disk group as a whole or the individual VxVM objects that comprise the disk group:

```

# vxstat -g dg_name -r
# vxstat -sr vol_name
# vxsta -pr vol_name

```

To view disk based statistics, run **vxstat -g dg\_name -d**, as shown in Example 8-5 on page 270.

#### Example 8-5 Disk level statistics from vxstat

---

```
# vxstat -g veritasdg -d
```

| TYP NAME     | OPERATIONS |       | BLOCKS |       | AVG TIME(ms) |       |
|--------------|------------|-------|--------|-------|--------------|-------|
|              | READ       | WRITE | READ   | WRITE | READ         | WRITE |
| dm veritas01 | 0          | 144   | 0      | 6164  | 0.0          | 20.8  |
| dm veritas02 | 0          | 144   | 0      | 6164  | 0.0          | 20.6  |
| dm veritas03 | 0          | 144   | 0      | 6164  | 0.0          | 19.6  |

---

The **vxstat -f f -g dg\_name**, can be used to display failed read or write operations.

The **vxtrace** utility can be run from the command line simply by giving it a volume name to monitor. This will trace all I/O for the selected volume. If the output is to be sent to a file, the **-d file\_name** option should be used. The resulting output file will be in binary format, readable by the **vxtrace** command. The **-o** option allows tracing of individual VxVM objects. Using the volume described in Example 8-2 on page 260, the **vxtrace** output in Example 8-6 was captured from standard output for the detach of a plex copy (**vxplex det testmirvol-01**).

#### Example 8-6 Sample vxtrace output

---

```
# vxtrace testmirvol
559 START write vol testmirvol op 0 block 16 len 2
559 END write vol testmirvol op 0 block 16 len 2 time 2
```

---

Following this, in a separate window, the **vxtstat** command was run at intervals to see the average time I/Os took in order to complete a plex attach for this volume (**vxplex att testmirvol testmirvol-01**) (see Example 8-7).

#### Example 8-7 vxstat output from a plex attach operation

---

```
# vxstat -fa -i2 -c20 testmirvol
...
Thu Aug 22 10:21:37 CDT 2002
vol testmirvol          59    14632   33.2

Thu Aug 22 10:21:39 CDT 2002
vol testmirvol          59    14632   33.9

Thu Aug 22 10:21:41 CDT 2002
vol testmirvol          60    14880   33.7

Thu Aug 22 10:21:43 CDT 2002
vol testmirvol          12     2840   32.5
```

---

Additionally, the AIX **iostat** command can be used to help identify potential I/O bottlenecks.

## Monitoring free space in a VERITAS File System

VERITAS file systems perform better when there is at least 10% free space available. The `df` command can be used to monitor free space. It is recommended that file system space be monitored and defragmentation carried out regularly with the `fsadm` command.

## 8.3 Tuning

VxVM and VxFS both have a number of global tuning parameters that can be modified. However, care should be taken, as there are some parameters that should not be modified. Prior to any tuning, the performance of the existing VxVM and VxFS configuration should be recorded, and the performance goals documented. Incremental changes should be made systematically to ensure system performance is not adversely affected by a change, and the impact assessed at each stage.

### 8.3.1 VxVM global parameters

A full list of the VxVM tunable parameters is shown in Example 8-8 (from the `smitty vxvm` fast path Change/Show VxVM Tunables menu). Changes will not take effect until the VxVM kernel extensions are reloaded either dynamically or at system reboot. For details of individual parameters please see “Tunable parameters” in Chapter 11, “Performance Monitoring and Tuning”, in the *VERITAS Volume Manager 3.2 - Administrator's Guide (AIX)*.

*Example 8-8 VxVM tuneables*

---

| Change/Show VxVM Tunables   |                |   |
|---|----------------|---|
| Type or select values in entry fields.<br>Press Enter AFTER making all desired changes. |                |   |
| [TOP]   | [Entry Fields] |   |
| Maximum number of subdisks per plex   | [4096]         | # |
| Maximum size of ioctl data (bytes)  | [32768]        | # |
| Maximum size of an I/O ioctl (sectors)  | [512]          | # |
| Maximum I/O size (sectors)  | [512]          | # |
| Maximum # of concurrent I/Os  | [4096]         | # |
| Default I/O delay for attaches (ticks)  | [50]           | # |
| Minimum DRL region size (sectors)   | [1024]         | # |
| Maximum # of DRL dirty regions (sectors)  | [2048]         | # |
| Maximum # of parallel vxconfigd I/O's allowed   | [256]          | # |
| Maximum total I/O trace buffer size (bytes)   | [4194304]      | # |
| Maximum size of each I/O trace buffer (bytes)   | [1048576]      | # |
| Default size of I/O trace buffer (bytes)  | [8192]         | # |

|  |            |           |          |
|--|------------|-----------|----------|
| Default size of error trace buffer (bytes)     | [16384]    | #         |          |
| Maximum # of trace channels                    | [32]       | #         |          |
| Default chkpt size (sectors)                   | [20480]    | #         |          |
| Maximum # of transient RSRs                    | [1]        | #         |          |
| VOLIOMEM Chunk size (bytes)                    | [65536]    | #         |          |
| Maximum memory allocated for each IOMEM pool   | [4194304]  | #         |          |
| Maximum memory allocated for VVR readback pool | [4194304]  | #         |          |
| Maximum memory allocated for VVR nmcom pool    | [4194304]  | #         |          |
| Minimum Threshold for in-memory updates        | [524288]   | #         |          |
| Transport used for VVR (1 for UDP, 2 for TCP)  | 1          | +         |          |
| FMR log size (kilobytes)                       | [4]        | #         |          |
| Maximum # of dirty regions in sequential mode  | [3]        | #         |          |
| F1=Help  | F2=Refresh | F3=Cancel | F4=List  |
| F5=Reset                                       | F6=Command | F7=Edit   | F8=Image |
| F9=Shell                                       | F10=Exit   | Enter=Do  |          |

### 8.3.2 VxFS global parameters

There are various VERITAS File System global tunable parameters that are set when VxFS is loaded. The default values calculated for these global tunables are based on the amount of memory available on the system. The default values do not change unless the amount of memory changes and VxFS is reloaded or the system is rebooted.

The `/etc/vx/vxfs` file contains VERITAS File System global tuning parameters. The `vxkextadm` utility loads the values for the tunables specified in `/etc/vx/vxfs` when loading the vxfs kernel extension. If the file contains an invalid entry, `vxkextadm` rejects the file and does not apply any of the tunable values.

**Note:** Kernel tunable parameters can degrade system performance if not applied correctly.

### 8.3.3 Self tuning file systems

When the VERITAS File System is used with VERITAS Volume Manager, the file system is automatically tuned at creation time. VERITAS Volume Manager is queried by the `mkfs` command when the file system is created to align the file system to the volume geometry.

As the volume geometry changes over time, VERITAS file systems on VERITAS volumes continue to be automatically tuned. The `mount` command also queries VERITAS Volume Manager when the file system is mounted and downloads the I/O parameters for the current geometry.



If the file system is being used with a hardware disk array or volume manager other than VERITAS Volume Manager, try to align the parameters to match the geometry of the logical disk. See 8.3.4, “File system tuning parameters” on page 273 for more information.

### 8.3.4 File system tuning parameters

VERITAS File System provides a set of tunable I/O parameters that control some of its behavior. These I/O parameters are useful to help the file system adjust to striped or RAID-5 volumes that could yield performance superior to a single disk. Typically, data streaming applications that access large files see the largest benefit from tuning the file system.

If the default global parameters (see 8.3.2, “VxFS global parameters” on page 272) are not acceptable, or the file system is being used without VERITAS Volume Manager, then the `/etc/vx/tunefstab` file can be used to set values for I/O parameters. The `mount` command reads the `/etc/vx/tunefstab` file and downloads any parameters specified for a file system. The `tunefstab` file overrides any values obtained from VERITAS Volume Manager. While the file system is mounted, any I/O parameters can be changed using the `vxtunefs` command, which can have tunables specified on the command line or can read them from the `/etc/vx/tunefstab` file.

The `vxtunefs` command can be used to print the current values of the I/O parameters for a given file system:

```
# vxtunefs -p mount_point
```

For an application to do efficient disk I/O, it should issue read requests that are equal to the product of `read_nstream` multiplied by `read_pref_io`. Generally, any multiple or factor of `read_nstream` multiplied by `read_pref_io` should be a good size for performance. For writing, the same rule of thumb applies to the `write_pref_io` and `write_nstream` parameters. When tuning a file system, the best thing to do is try out the tuning parameters under a real life workload.

If an application is doing sequential I/O to large files, it should try to issue requests larger than the `discovered_direct_iosz`. This causes the I/O requests to be performed as discovered direct I/O requests, which are unbuffered like direct I/O but do not require synchronous inode updates when extending the file. If the file is larger than can fit in the cache, using unbuffered I/O avoids removing useful data out of the cache and lessens CPU overhead.

## 8.4 Application interface support

The VERITAS File System (VxFS) provides an interface for applications to utilize various cache advisories. This section briefly describes those cache advisories that can be used to enhance performance. Application use of extent based information is also available via application interface support and is commented on in 8.1.3, “Extent-based allocation” on page 263.

### 8.4.1 Cache advisories

Two interfaces exist for implementing cache advisories. They may be requested for a specific file under program control, or they may be implemented for all files on a file system through the **mount** option **mincache**. Use of a **mount** option allows the benefits of cache advisories to be provided to existing applications without code changes or recompilation.

Cache advisories are set within a program by calling the **VX\_SETCACHE ioctl** command. The current set of advisories can be obtained with the **VX\_GETCACHE ioctl** command. For details on the use of these **ioctl** commands, see the **vxfsio(7)** manual page.

#### Data synchronous I/O

While synchronous I/O operations require that data and the meta data structures needed to access them be written to disk, it is not always necessary to flush all meta data synchronously. For example, while a write that extends its file should update the data and the size and pointer information synchronously (ensuring that the data is not lost), it is not necessary to update time stamps synchronously for every access or non allocating modification. VERITAS File System supports a data-synchronous mode, which updates data and allocation meta data synchronously while buffering meta data updates that have no effect on data accessibility.

This can be used to accelerate synchronous, file-based applications. This can include personal databases, such as Ingres and uniVerse, which do not support their own block buffering, but which may be accelerated somewhat by not requiring a synchronous inode update for each access.

#### Direct I/O

In direct mode, VxFS does not copy data between user and kernel buffers. Instead, it performs file I/O directly into and out of user buffers. This optimization, with very large extents, allows file accesses to operate at raw-disk speed. Direct I/O mode always is enabled safely. I/O requests that do not meet page alignment requirements, or that might conflict with mapped I/O requests to the same file, are performed as data-synchronous I/O. Direct I/O may be enabled via a

programmatic interface or via a **mount** option. For DBMS engines, this can provide the benefits of raw disk speed with the administrative ease of a file system.

### **Discovered direct I/O**

For larger I/Os, direct I/O can perform much better than buffered I/O. For small writes, buffered I/O generally performs much better than Direct I/O. For environments where there is a mixed I/O size, it is ideal to switch between the two methodologies to maximize performance. VxFS implements discovered direct I/O that provides a user-definable watermark. When the file system receives an I/O larger than this watermark, it will use direct I/O. All I/Os smaller than or equal to the watermark are buffered.

## **8.4.2 Other programatic advisories**

Two additional advisories, **VX\_SEQ** and **VX\_RANDOM**, are set using the **VX\_SETCACHE ioctl** command from within a program. They are used to clearly set the access mode of the I/O to gain improved performance. For details on the use of VERITAS **ioctl** commands, see the **vxfsio(7)** manual page.

### **VX\_SEQ advisory**

The **VX\_SEQ** advisory indicates that the file is being accessed sequentially. When the file is being read, the maximum read-ahead is always performed. When the file is written, instead of trying to determine whether the I/O is sequential or random by examining the write offset, sequential I/O is assumed. The pages for the write are not immediately flushed. Instead, pages are flushed some distance behind the current write point.

### **VX\_RANDOM advisory**

The **VX\_RANDOM** advisory indicates that the file is being accessed randomly. For reads, this disables read-ahead. For writes, this disables the flush-behind. The data is flushed by the pager, at a rate based on memory contention.

## **8.5 Scalability**

Scalability refers to the ability of software to adapt readily to a greater or lesser intensity of use, volume, or demand while still meeting business objectives. Software scalability can be evaluated from several viewpoints:

- ▶ Architectural scalability
- ▶ Administrative scalability
- ▶ Scaling services

## 8.5.1 Architectural scalability

Architectural or design scalability refers to software’s ability to continue to provide optimal performance as demands increase. VERITAS Foundation Suite has several design features that allow it to grow with your server environment.

### Version 4 disk layout

The VERITAS File System Version 4 disk layout allows the file system to scale easily to accommodate large files and large file systems.

The VERITAS File System Version 4 disk layout divides the entire file system space into fixed size allocation units. The first allocation unit starts at block zero and all allocation units are a fixed length of 32 KB blocks. (An exception may be the last AU, which occupies whatever space remains at the end of the file system.) Because the first AU starts at block zero instead of part way through the file system, as in previous versions, there is no longer a need for explicit AU alignment or padding to be added when creating a file system.

The Version 4 file system also moves away from the model of storing AU structural data at the start of an AU and puts all structural information in files. So expanding the file system structures simply requires extending the appropriate structural files. This removes the extent size restriction imposed by the previous layouts.

### VERITAS volume manager limits

Table 8-1 to Table 8-3 on page 277 are included in the comparison whitepapers referenced in 6.2, “AIX LVM, JFS/JFS2 and VxVM, VxFS compared” on page 193. VERITAS Volume Manager offers apparent increased scalability over the AIX Logical Volume Manager (LVM), although the rapid growth of commercial disk capacities is such that these limits are unlikely to be reached.

Table 8-1 Volume group limits

| Parameter  | LVM                                    | VxVM       |
|--|--|------------|
| Equivalent   | Volume group                           | Disk group |
| Limits   | 32-bit: 256<br>64-bit: 4096 (see Note) | None       |
| Note: 4096 is the LVM architected limit, but the actual maximum is less than 1024 Volume groups due to the device table limitation of 1024 active devices using the 64-bit kernel. |  |            |

Table 8-2 Physical volumes in volume/disk group

| Parameter  | LVM                              | VxVM                       |
|------------|----------------------------------|----------------------------|
| Equivalent | Physical volumes in volume group | Disk access and disk media |
| Limits     | Standard: 256 or Big: 512        | None                       |

Table 8-3 Mirror copies per logical volume

| Parameter  | LVM                          | VxVM            |
|------------|------------------------------|-----------------|
| Equivalent | Copies per logical partition | Copies per plex |
| Limits     | 3                            | 32              |

## 8.5.2 Administrative scalability

The administrative tools within VERITAS Foundation Suite allow for administrative scalability. Many typical administrative tasks are either made easier or automated.

### File system tuning

VERITAS file systems on VERITAS volumes are automatically tuned by aligning file system parameters with the underlying disk geometry at creation time. Each time the file system is mounted, the disk geometry is reanalyzed and mount parameters are tuned for the current geometry. This eliminates the need to manually tune file system parameters as the disk geometry changes over time. See 8.3.3, “Self tuning file systems” on page 272 for more information.

### VERITAS Enterprise Administrator (VEA)

The VERITAS Enterprise Administrator (VEA) enables systems administrators to remotely manage VERITAS Foundation Suite on multiple servers from a single administrative interface.

### Single method of management

In heterogeneous environments, VERITAS Foundation Suite provides the basis for a single method of storage management across the most prevalent UNIX platforms. A single set of management procedures and a common administrative interface reduces training, streamlines procedures, and allows administrators to be more efficient.

### 8.5.3 Scaling services

VERITAS Foundation Suite can enhance the performance of applications using VERITAS file systems and VERITAS volumes. Some must be implemented within the program, others as **mount** options, and still others as add-on options to VERITAS Foundation Suite.

#### Cache advisories

Cache advisories can be implemented either within a program or as a mount option. Cache advisories tailor the I/O access method to increase application performance. See 8.4, “Application interface support” on page 274 for a more detailed description.

#### VX\_SEQ and VX\_RANDOM

VX\_SEQ and VX\_RANDOM are two additional advisories to further tune application I/O. Unlike cache advisories, which can be specified as mount options, VX\_SEQ and VX\_RANDOM can only be implemented from a program via the **VX\_SETCACHE ioctl** command. See 8.4.2, “Other programmatic advisories” on page 275 for additional information.

#### VERITAS Database Editions

VERITAS Database Editions are integrated suites of data and storage management technologies designed to optimize the performance of supported databases.

# Troubleshooting and technical support

This chapter discusses troubleshooting and information for VERITAS Foundation Suite for AIX.

In this chapter, we will discuss:

- ▶ How to get patches
- ▶ How to get technical support
- ▶ Installation issues
- ▶ Administration issues
- ▶ References for troubleshooting

## 9.1 How to get patches

In this section, we describe how to obtain patches or fix updates from VERITAS and IBM.

### 9.1.1 How to get patches from VERITAS

To obtain fix updates and patches to VERITAS Foundation Suite for AIX, you need to access VERITAS Software's main support Web site at:

<http://support.veritas.com>

See Figure 9-1 shows the Web site.

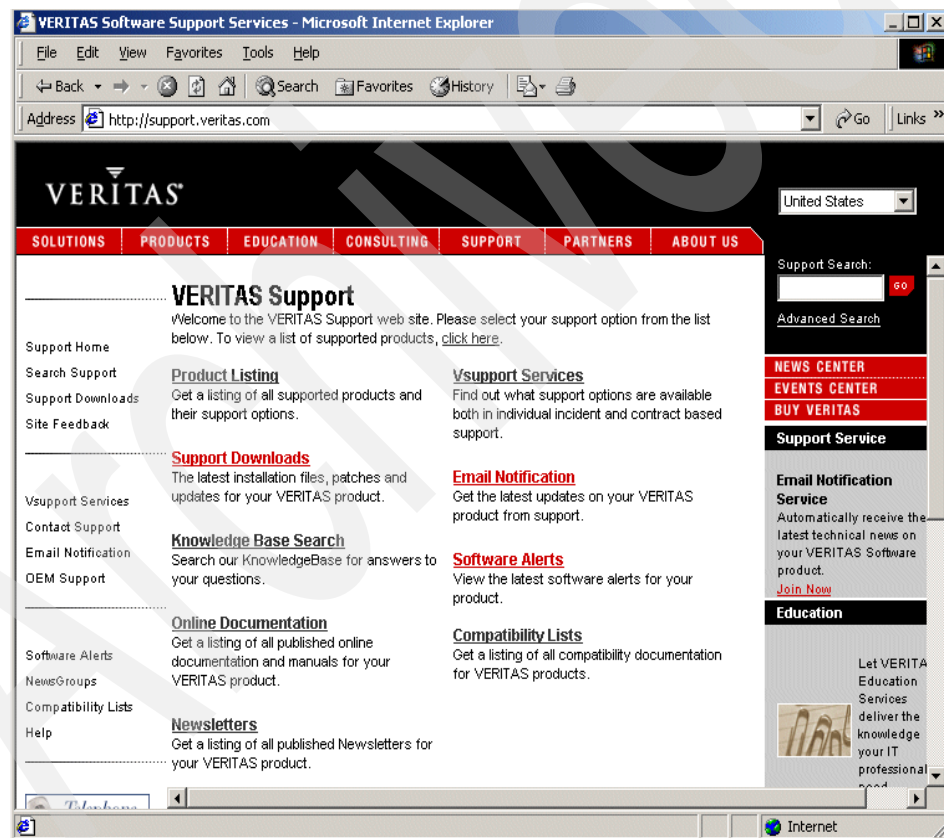


Figure 9-1 VERITAS Support Web site

From this Web site, you can sign up for e-mail notification of VERITAS TechAlerts, which will include information on the latest patches and fix updates.



To sign up, click on **Email Notification**. Choose **TechAlerts** as the type of notification that you wish to receive. After typing in your e-mail address and your country location, you will be taken to a page that lists available products. At a minimum, sign up for tech alerts for **File System for UNIX** and **Volume Manager for UNIX**, which will provide information on VERITAS File System and VERITAS Volume Manager for AIX. Add any other products about which you wish to receive alerts. See Figure 9-2 for an overview of this process.

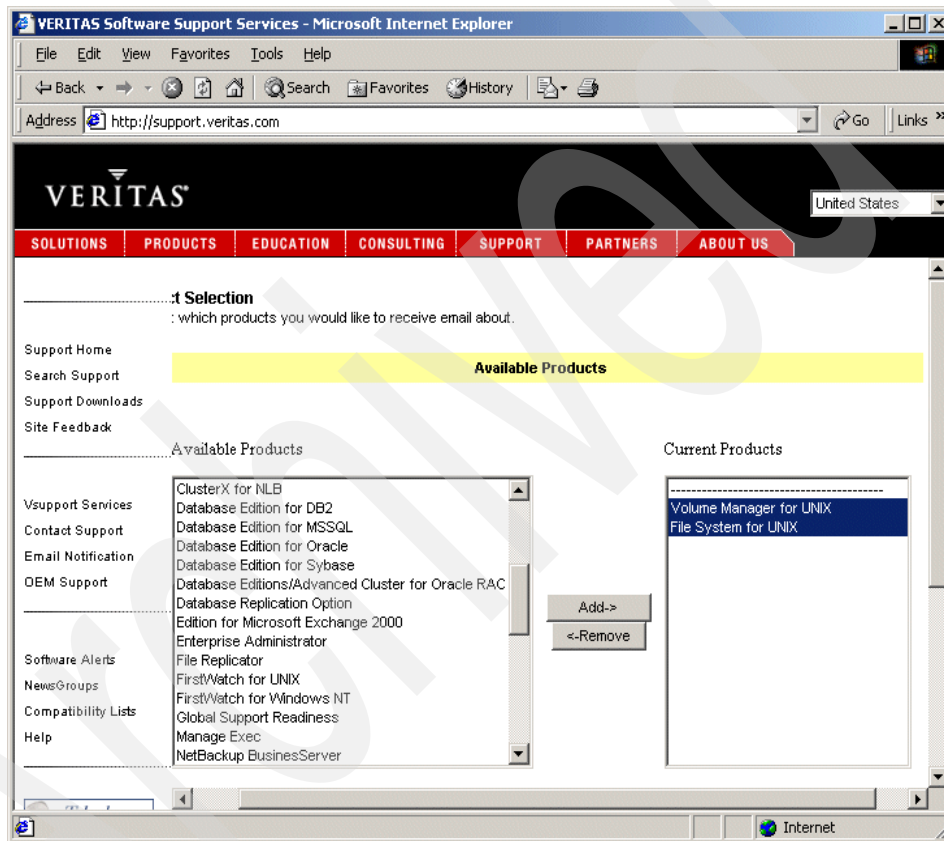


Figure 9-2 VERITAS TechAlerts Product Selection page

You can also look for the latest VERITAS alerts by clicking on **Software Alerts** from the main VERITAS support Web site at <http://support.veritas.com>, or you can access it directly at:

[http://support.veritas.com/prodlist\\_path\\_alerts.htm](http://support.veritas.com/prodlist_path_alerts.htm)

You can check here periodically for new tech alerts. Available patches and fix updates will be announced at this Web site. To get information regarding VERITAS Foundation Suite for AIX, click on **Foundation Suite**, and then click on

either **File System for UNIX** or **Volume Manager for UNIX**. At this point, you will see the page shown in Figure 9-3. Where it says **Select a platform:**, choose **AIX**. You will then see relevant information about VERITAS File System or VERITAS Volume Manager on AIX 5L. Clicking on the individual documents will provide detailed information, and if a patch is available, you will see how to download the patch from the VERITAS ftp site.

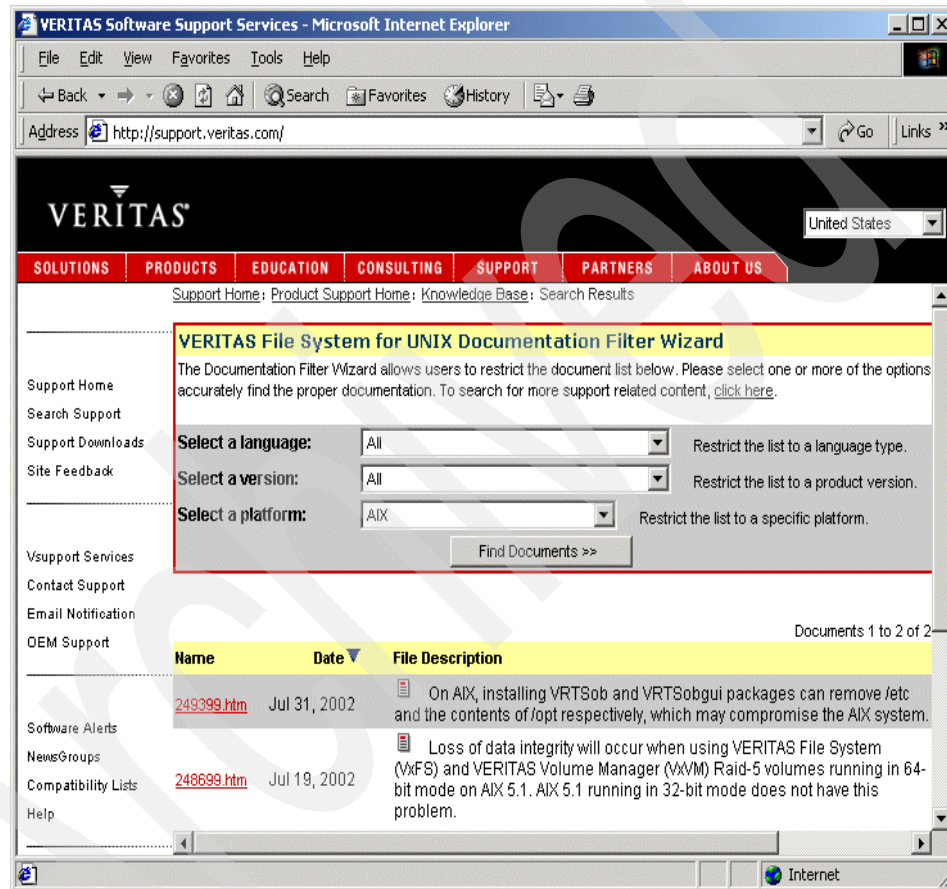


Figure 9-3 VERITAS File System for UNIX page containing AIX tech alert

## Available patches

At the time of writing this book, there are three patches available that pertain to VERITAS Foundation Suite for AIX. See 9.3.1, “VERITAS patches” on page 290 for information on these patches.

## 9.1.2 How to get patches from IBM

Fix updates and patches for IBM AIX 5L Version 5.1 are available for download directly from the following Web site:

<http://techsupport.services.ibm.com/server/aix.fdc51>

### Fileset updates

You can also download individual filesets from the same Web site. For example, if you need to get the latest bos.rte.commands fileset prior to installing VERITAS Foundation Suite, then you would go to the above Web site and type in bos.rte.commands in the main search window. Figure 9-4 shows the main IBM technical support Web site page for IBM @server pSeries AIX 5L.

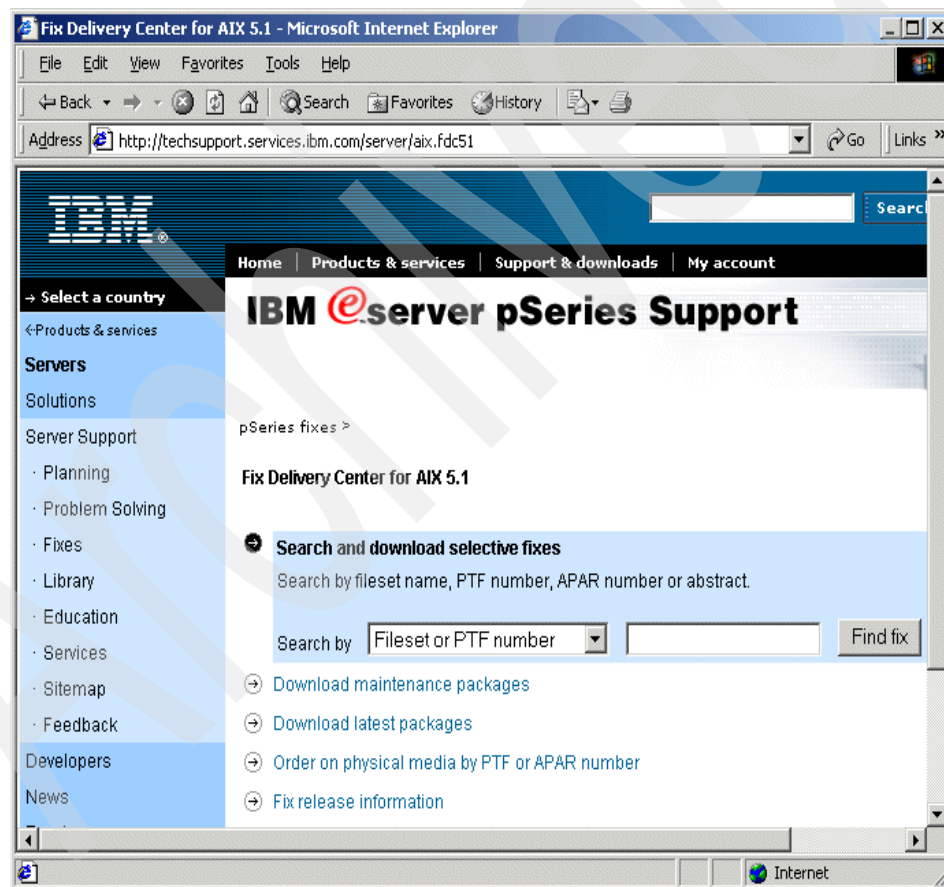


Figure 9-4 IBM eServer pSeries AIX 5L Technical Support Web site

On the next page (Figure 9-5), you will see a list of available versions for that fileset. Select the version that you need, and click on **Add to list**, and then **Proceed to packaging**.

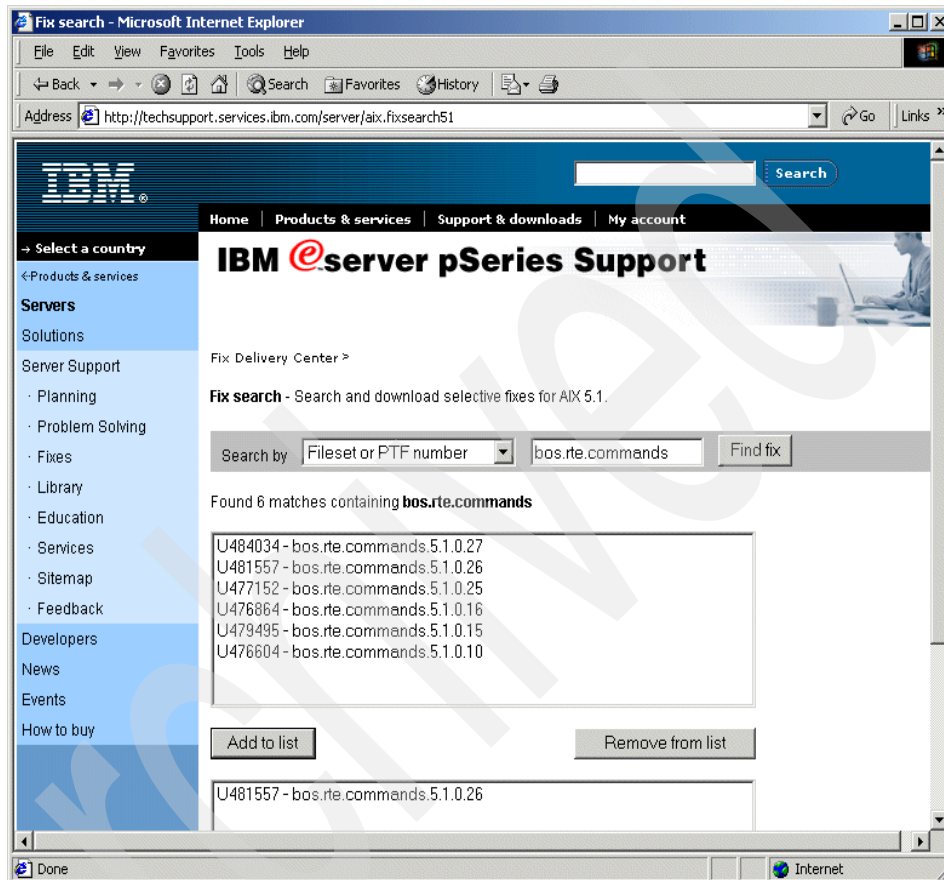


Figure 9-5 Select AIX 5L fileset version to download

After selecting the fileset fix update that you need, then you will get a page where you need to indicate what maintenance level is installed. On the last page, you will have the option to download the update through either HTTP or FTP protocols. The fileset will be in the .bff format, which can be installed by using SMIT or the **smitty install\_update** fast path.

## 9.2 How to get technical support

In this section, we discuss how to obtain technical support information from both VERITAS and IBM.

### 9.2.1 How to get technical support from VERITAS

Technical support for VERITAS Foundation Suite for AIX is available directly from VERITAS. In the United States, you can call VERITAS at the toll-free telephone number 800-342-0652 or at the regular number 407-531-7600. Support is also available via e-mail by writing to [support@veritas.com](mailto:support@veritas.com). For complete information on accessing technical support, visit the VERITAS support Web site at <http://support.veritas.com>. A full list of worldwide technical support phone numbers is available from the VERITAS support Web site, as well as an e-mail technical support form.

Another method of getting technical support information is to make use of the VERITAS Knowledge Base, also accessible from <http://support.veritas.com>. This is a good place to start to look for answers to technical questions. Click on **Knowledge Base Search** to begin.

There is also a list of new VERITAS alerts available at:

[http://support.veritas.com/prodlist\\_path\\_alerts.htm](http://support.veritas.com/prodlist_path_alerts.htm)

You can check this Web site periodically to see if there are any relevant alerts that you should be aware of. To get information regarding VERITAS Foundation Suite for AIX, click on **Foundation Suite**, and then click on either **File System for UNIX** or **Volume Manager for UNIX** to see a list of the relevant tech alerts.

#### Newsgroups

There are also many newsgroups devoted to VERITAS products. A list of the newsgroups is available on the VERITAS support Web site. These newsgroups, however, are not maintained or supported in any way by VERITAS. They are outside public newsgroups that provide for the exchange of information between VERITAS users. To access the newsgroups, go to the VERITAS support Web site at <http://support.veritas.com>, where you can find a list of the newsgroups by clicking on **Newsgroups** on the left navigation bar. Some of the relevant newsgroups are:

- ▶ [veritas.file\\_system.english](#)
- ▶ [veritas.volume\\_manager.english](#)

You can also access the newsgroups by going to:

<http://news.support.veritas.com>

## Mailing lists

It is a good idea to sign up for the e-mail notification service. You can customize the type of VERITAS alerts that you would like to receive. Sign up by going to <http://support.veritas.com> and clicking on **Email Notification**, or by going directly to <http://maillist.support.veritas.com/subscribe.asp>. You can then choose to sign up for the following types of e-mail notifications:

- ▶ **TechAlerts:** This will send e-mail regarding critical technical issues. VERITAS strongly encourages you to at least sign up for this e-mail alert.
- ▶ **Digests:** This will send an e-mail digest regarding all technical documents and file publications periodically, usually once a month. VERITAS encourages system administrators and others using VERITAS software on a daily basis to sign up for this e-mail alert.
- ▶ **Newsletters:** This will send a monthly e-mail newsletter, which includes best practices and other technical information that can be useful for administering and using VERITAS software.
- ▶ **Beta Programs:** This will send an e-mail telling about available beta programs. You can apply directly for the beta programs at:  
<http://beta.veritas.com>
- ▶ **Services:** This will send e-mail regarding customer education, consulting services, and VSupport Service offerings.

Once you have chosen the categories that you are interested in, you will be taken to a Web page where you can select the VERITAS products that you are interested in. On this page, to get information on VERITAS Foundation Suite for AIX, you will need to select both **File System for UNIX** and **Volume Manager for UNIX**.

## How to prepare for a technical support call with VERITAS

The VRTSExplorer program can be used by VERITAS technical support to determine the cause of any technical support issue that may arise. It is recommended that you install the VRTSExplorer program and run it prior to initiating a technical support call with VERITAS. You can download this program from the VERITAS ftp site or install it from the VERITAS Foundation Suite for AIX CD or the AIX 5L Bonus Pack CD. VRTSExplorer does not require any additional licenses. Some information on installing VRTSExplorer is "Installing VRTSExplorer" on page 71. Let us go through the installation and usage of VRTSExplorer in more detail.



### ***Downloading VRTSexplorer from the ftp site***

1. In a Web browser, go to:  
<ftp://ftp.veritas.com/pub/support/vxexplore.tar.Z> and download the file, or use the ftp program and follow these steps:

```
open ftp.veritas.com
login anonymously
cd /pub/support
bin
get vxexplore.tar.Z
bye
```

Save the file to any location on your AIX 5L machine.

2. Log in as root and uncompress and untar the file:

```
# uncompress vxexplore.tar.Z
# tar xvf vxexplore.tar
```

Be aware that the tar will untar the VRTSexplorer in whatever directory the vxexplore.tar file resides, so if you want to put the VRTSexplorer in a particular directory, also put the tar file there.

3. Run the VRTSexplorer program located in the VRTSexplorer directory:

```
# ./VRTSexplorer/VRTSexplorer
```

### ***Installing VRTSexplorer from the CD***

The VRTSspt file is included on the VERITAS CD under the /support directory. To load the software from CD-ROM:

1. Log in as root.
2. Mount the CD on the mount point:

```
# mount -V cdrfs -o ro /dev/cd0 /cdrom
```

3. Change directory to the support directory and install the VRTSspt package:

```
# cd /cdrom/support
# installp -ac -d VRTSspt.bff VRTSspt
```

4. The program is installed in the /opt/VRTSspt directory. Run the program:

```
# /opt/VRTSspt/VRTSexplorer/VRTSexplorer
```

### ***Creating the VRTSexplorer information tar file***

1. VRTSexplorer will prompt for a destination directory for the information that it collects into a compressed tar file. Press Return to accept the default directory or enter any other directory name that you want. You will see some messages. You will also get a warning that VRTSexplorer will stop the VxVM configuration daemon (vxconfigd). You have the opportunity to abort the VRTSexplorer at this point, if you need to. Otherwise, answer y and hit Return. VRTSexplorer writes the results of its investigations to a compressed

tar file named VRTSexplorer\_casenumbr\_hostname.tar.Z in the directory you specified. It is quick to run VRTSexplorer; it should not take more than a few minutes. The tar file that is generated will contain a huge number of files, including information on:

- system
  - boot
  - cron
  - physical devices
  - **df** output
  - etc files
  - **df -k** output
  - instfix
  - licenses
  - LVM physical volumes, volume groups
  - networks
  - **ps** output
  - smit logs
  - VxVM volumes, disk groups
2. Once VRTSexplorer has generated the compressed tar file, upload the file with either your Web browser or the ftp program to the VERITAS Technical Support anonymous incoming ftp site at:  
<ftp://ftp.veritas.com/incoming>
  3. The next step is to notify VERITAS that you have uploaded the VRTSexplorer compressed tar file. Either call VERITAS Technical Support at 1-800-258-8649 (in the U.S.) and give the name of the uploaded tar file, or you can e-mail [support@veritas.com](mailto:support@veritas.com) with the same information. If you have a case ID number already assigned, include that in the subject line of your e-mail.

## 9.2.2 How to get technical support from IBM

You should first go to VERITAS for technical support for VERITAS Foundation Suite for AIX, but if you have a technical support issue with AIX 5L, then here are some Web sites that contain technical support information.

### General technical support information

Comprehensive technical support information is available by going to IBM's main Web site at <http://ibm.com> and then clicking on **Support & downloads**. Drill down to IBM @server pSeries support.



## Tech tips for system administrators

You can get technical support information from several Web sites off of <http://ibm.com>. The first Web site is a **Tips for AIX administrators**, available at <http://techsupport.services.ibm.com/server/aix.techTips>. Select **AIX 5L 5.1** and then you can see information in these categories:

- ▶ AIX 5L 5.1
- ▶ Advisories
- ▶ Announcements
- ▶ Miscellaneous
- ▶ Tools

There is a lot of useful information available from here, so take a look.

## Newsgroup

The best newsgroup to look for information regarding AIX 5L questions is the comp.unix.aix newsgroup. You can either search the comp.unix.aix newsgroup from any Internet search engine, or you can find information regarding the newsgroup at the IBM @server pSeries AIX 5L Tips and how-to page, available at <http://techsupport.services.ibm.com/server/nav?fetch=tah>.

## AIX 5L troubleshooting

Troubleshooting information for AIX 5L is available at <http://techsupport.services.ibm.com/server/nav?fetch=ts>. This Web site contains general AIX Authorized Problem Analysis Reports (APARs), which are known issues, both open and closed. The release notes for AIX 5L are also on this Web site, as well as the README files.

## Mailing lists

You can sign up for software technical mailings at <http://techsupport.services.ibm.com/server/listserv>. Included are mailings on:

- ▶ Install Tips - AIX 5L 5.1
- ▶ Preventive Maintenance - AIX 5L 5.1
- ▶ CERT Advisories - Current
- ▶ Lists of AIX O/S Fixes

Hardware technical mailings that e-mail you information on the latest microcode updates are also available. Go to <http://techsupport.services.ibm.com/rs6k/mcode.html> for more information.

## 9.3 Installation issues

In this section, we note installation issues that you may encounter and also installation prerequisites that you should be aware of. Many of these issues have been addressed in later versions of the software, so only earlier versions, including the AIX 5L July 2002 Bonus Pack, may see the items described here. Also, most of these issues are documented in the installation procedures described in Chapter 3, “Planning and installation” on page 41. They are included here for easy access.

### 9.3.1 VERITAS patches

At time of writing this redbook, patches are available from VERITAS for the following issues that pertain to AIX:

1. On AIX, installing VRTSob and VRTSobgui packages can remove /etc and the contents of /opt respectively, which may compromise the AIX system. The TechNote on this issue from VERITAS provides the following information:

“VERITAS Enterprise Administrator (VEA) contains two packages, VRTSob and VRTSobgui. Installing VRTSob can remove the /etc directory, and installing VRTSobgui can remove the contents of the /opt directory. As a result, the AIX system may become unusable, and the /etc and /opt directories will need to be restored from backup. If the VRTSob and/or VRTSobgui packages are already installed, and the /etc and /opt directories are intact, then no immediate action is required.

This problem occurs *only* if *both* of the following conditions coexist:

- The **inurid -r** command has been executed
- Installing VRTSob or VRTSobgui versions 3.0.255.0 or 3.0.255.1”

The **inurid -r** command removes information used for installation of diskless/dataless clients from the inst\_root directories of installed software.

For more information on this issue, see the VERITAS TechNote ID: 249399. To install the patch to resolve this issue, see 9.1.1, “How to get patches from VERITAS” on page 280.

2. Loss of data integrity will occur when using VERITAS File System (VxFS) and VERITAS Volume Manager (VxVM) RAID 5 volumes running in 64-bit mode on AIX 5L 5.1. AIX 5L 5.1 running in 32-bit mode does not have this problem. To solve this, at time of writing this book, please follow the information provided below in the TechNote from VERITAS:

“A VxFS file system must *not* be used with VxVM RAID 5 volumes. Loss of data integrity occurs when data is flushed from memory to disk. A VxFS file system may be used with all other VxVM volume layouts, except RAID 5.

This problem occurs *only* if ALL following conditions coexist:

- VRTSvxfs fileset level is 3.4.2.0 or 3.4.2.1.
- AIX 5L 5.1 is running in 64-bit mode.
- A VxFS file system is created on a VxVM RAID 5 volume.

The problem occurs *only* on systems running in 64-bit mode on AIX 5L 5.1.

The problem does *not* exist in 32-bit mode on AIX 5L 5.1.

To verify your version of VxFS, run:

```
# lspp -l VRTSvxfs
```

To verify if 64-bit mode is enabled, run:

```
# ls -la /unix
```

If /unix is linked to /usr/lib/boot/unix\_64, then the 64-bit kernel is enabled.

Please refer to IBM TechDoc 97544978625424 for more information on how to check whether 64-bit mode is enabled (see <http://techsupport.services.ibm.com>).

To verify if VxVM volume is RAID 5, run:

```
# vxprint -th -g <dgname> <volname>
```

Until the problem is resolved, do *not* use VxFS file systems with VxVM RAID 5 volumes.

Until VERITAS releases a permanent fix, do not use VxVM RAID 5 volumes on AIX 5L 5.1 running in 64-bit mode. If redundancy is required, then create VxVM mirrored volumes, for example, concat-mirror or stripe-mirror. A permanent fix will be available in the next VxFS 3.4 patch, release level 3.4.2.2. VxFS 3.4.2.2 is estimated to be released the first week of September 2002. VxFS 3.4.2.2 will completely resolve the issue with RAID 5 volumes.”

For more information on this issue, see the VERITAS TechNote ID: 248699. See 9.1.1, “How to get patches from VERITAS” on page 280 for information on downloading the patch.

3. There is a possible loss of data integrity during snapback of snapshot volumes if Fast Mirror Resynchronization (FMR) is enabled. The following TechNote information is provided by VERITAS:

“Loss of data integrity during snapback of snapshot volumes is possible if Fast Mirror Resynchronization is enabled. Fast Mirror Resynchronization (FMR), also known as FastResync and part of FlashSnap, is a feature in VERITAS Volume Manager (VxVM) that increases the performance of snapshot operations. With FMR enabled, if data is written to the original volume at the exact same time the snapshot is being taken, loss of data integrity may occur during the snapback operation.

This problem occurs *only* if *all* following conditions coexist:

- VERITAS File System (VxFS) is used on the volume.
- Either Non-Persistent FastResync (introduced in VxVM 3.1) or Persistent FastResync (introduced in VxVM 3.2) is enabled on the volume.
- Data is written to the original volume when the **snapshot** command was issued.
- The snapshot volume has been snapped back.”

This issue exists on all UNIX platforms, including SUN Solaris, HP-UX, and AIX. More information on this issue is available from VERITAS in the TechNote ID: 249402. See 9.1.1, “How to get patches from VERITAS” on page 280 for more information on downloading the patch for this issue.

### 9.3.2 IBM APARs

Prior to starting an installation of VERITAS Foundation Suite for AIX, be sure you have installed the necessary APARs on your AIX 5L Version 5.1 machine. See Chapter 3, “Planning and installation” on page 41 for a list of the needed APARs. See 9.1.2, “How to get patches from IBM” on page 283 for more information obtaining these APARs. Also see <http://seer.support.veritas.com/docs/245873.htm>, which documents required AIX filesets needed prior to installing VERITAS Foundation Suite HA Version 3.4.

### 9.3.3 Possible installation issues

1. Ensure that you have enough file system space prior to starting an installation. The requirements are listed in 3.1.3, “File system space” on page 45. Although the file systems should be extended automatically as needed during the installation process, the tested versions did not do this. This issue is being addressed and will be fixed in the next release.
2. When running the VRTSinstall on an AIX 5L machine installed with a 32-bit operating system (versus a 64-bit operating system), you may see a message that the bos.64bit and bos.mp64 filesets have not been installed. This does not affect the installation and you can safely ignore these messages. Later releases of VERITAS Foundation Suite for AIX fix this issue. See 3.1.3, “File system space” on page 45 for details on the messages.
3. When installing via VRTSinstall, at the main VRTSinstall window you must either select ‘all’ packages to install or select each package individually. It is not possible to make multiple selections, such as ‘1,2,3,4’. See Example 9-1 on page 293 for details. This issue will be fixed in later releases.

### Example 9-1 VRTSinstall main window, multiple selections not allowed

---

```
*****
Veritas Foundation Suite Installation
*****
```

1. VRTSfsdoc
2. VRTSfspro
3. VRTSob
4. VRTSobgui
5. VRTSvlic.2.10.004
6. VRTSvlic.2.10.4.2
7. VRTSvm doc
8. VRTSvmman
9. VRTSvmpro
10. VRTSvxfs
11. VRTSvxvm

```
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [(1-11),?,??,q]:1,2,3,4
./VRTSinstall[373]: 1,2,3,4: 0403-057 Syntax error
```

---

4. When using **vxinstall** to change DMP options and selecting option 4 "Suppress all but one path to a disk" and then entering a disk that does not exist, it returns with the following output:  
Input should be all paths (c#t#d#)  
This is a bit confusing, because the suggested disk name is not consistent with AIX. This will be fixed in later releases.
5. After installing VRTSob, it leaves a 28 MB tar file called jre.tar in the /opt/VRTSob directory. After installation of VRTSob, this file is not needed and can be safely deleted.
6. When installing the VRTSob and VRTSobgui packages, it extracts the jre.tar tar file mentioned above. If there is not enough space to extract this tar file, it still reports the status of the package installation as being successful, even though there will be errors in the smit.log, as shown in Example 9-2. If you see this error, then manually increase the size of the file system and redo the installation.

### Example 9-2 VRTSob and VRTSobgui package installation failure

---

```
copyright notice for VRTSvcsvr >>. . . .
Filesets processed: 6 of 11 (Total time: 1 mins 7 secs).
installp: APPLYING software for:
VRTSobgui 3.0.255.0
tar: 0511-197 jre/lib/i18n.jar: Cannot write data extracted with the tar
command: There is not enough space in the file system.
Filesets processed: 7 of 11 (Total time: 1 mins 13 secs).
```

```
installp: APPLYING software for:
VRTSob 3.0.255.0
Installation Summary
-----
```

| Name      | Level     | Part      | Event | Result  |
|-----------|-----------|-----------|-------|---------|
| VRTSvxvm  | 3.2.0.0   | USR       | APPLY | SUCCESS |
| VRTSvxvm  | 3.2.0.0   | ROOT      | APPLY | SUCCESS |
| VRTSvxfs  | 3.4.2.0   | USR       | APPLY | SUCCESS |
| VRTSvxfs  | 3.4.2.0   | ROOT      | APPLY | SUCCESS |
| VRTSvmman | 3.2.0.0   | USR       | APPLY | SUCCESS |
| VRTSvmman | 3.2.0.0   | ROOT      | APPLY | SUCCESS |
| VRTSvmdoc | 3.2.0.0   | USR       | APPLY | SUCCESS |
| VRTSvmdoc | 3.2.0.0   | ROOT      | APPLY | SUCCESS |
| VRTSvlic  | 2.10.4.0  | USR       | APPLY | SUCCESS |
| VRTSobgui | 3.0.255.0 | USR       | APPLY | SUCCESS |
| VRTSobgui | 3.0.255.0 | ROOT      | APPLY | SUCCESS |
| VRTSob    | 3.0.255.0 | USR       | APPLY | SUCCESS |
| VRTSob    |           | 3.0.255.0 | ROOT  |         |
| APPLY     | SUCCESS   |           |       |         |

7. To access the VERITAS Foundation Suite man (manual) pages after installation has completed, it is necessary to add the VERITAS man pages directory to the MANPATH environment variable. In a Korn or Bourne shell, you can do this by executing the following command:

```
# export MANPATH=$MANPATH:/opt/VRTS/man
```

In a C shell:

```
# setenv MANPATH $MANPATH:/opt/VRTS/man
```

8. The only daemon running after installation has completed will be the **vxconfigd** daemon. Note that you need to enable the **vxconfigd** daemon by running:

```
# /usr/sbin/vxdctl enable
```

9. Some installation information regarding the success or failure of package installation is available in the /tmp/VRTSinstall.out file. This is not a complete log, but you can look here for some information.
10. For VxFS, Oracle failed to create database files on the VxFS file system for VxFS Version 3.4.2.0. This error occurs only if the block size is set to 8 KB. This is fixed in VxFS Version 3.4.2.1.
11. A shutdown script does not currently exist for AIX. The shutdown script is not actually required, but will be added in later releases. It does some tasks, such as setting vxiods back to previous values, as well as making sure that the file system is shut down cleanly. The script exists on other UNIX platforms.

## 9.4 Administration issues

1. Prior to creating any disk groups, especially in a networked environment, ensure that you have set the host name to a unique name. Do not use localhost or a host name that exists in your networked environment. The host name is used during disk group creation as part of identifying the disk group, and if set to a non-unique name, this could cause conflicts with disk groups on other machines sharing the same host name.
2. If you need to convert LVM volumes to VxVM, then you can use the **vxvmconvert** command. But if you need to convert JFS file systems to VxFS, then there does not yet exist on AIX a similar **vxfsconvert** command. The **vxfsconvert** command is planned, but is not yet available. However, a conversion utility is available from VERITAS services. Contact your VERITAS account manager if you require a JFS to VxFS conversion. There is no support for JFS2 to VxFS conversion at the time of writing this book. See Chapter 7, “Migration considerations” on page 219 for more information.
3. To change your VERITAS disk groups and the data in the disk groups back to native AIX volume groups with the data intact, there is no tool available. You can do this by using normal UNIX commands, such as **dd**, **tar**, and so on, to back up the data and then repopulate the data in the native AIX volume groups.
4. Note that disks are required for both rootdg and rootvg. This is necessary because “rootability” is not available on AIX at the time of writing. If you are concerned about this being a waste of space, you could put extra swap space or a non-critical file system in rootdg to make better use of that space.
5. In the current release, it is not possible to extend a VERITAS file system if it is completely full. This constraint had existed on other UNIX platforms and has been fixed elsewhere, but this fix for AIX will be available in the next release. To do the VxFS file system increase, you must free up at least 50 MB; otherwise, it will return with an `errno 28`. You may be able to increase the underlying volume, but you will not be able to increase the VxFS file system until more space is available. Again, a fix is available in the next release.
6. When renaming a disk group, it is necessary to manually edit the `/etc/filesystems` file. In Example 9-3 on page 296, we want to rename the disk group called olddg to newdg. There is already a volume on this disk group called vol01. Once we rename the disk group, we need to manually edit `/etc/filesystems` so that the stanza that contains the VxFS file system mounted on the disk group points to the correct volume. The issue of requiring a manual edit of the `/etc/filesystems` file will be addressed in the next release.

### Example 9-3 Renaming a disk group and editing /etc/filesystems

---

```
# vxdg -n newdg deport olddg
# vxdg import newdg
# mount -v vxfs /testfs
vxfs mount: Cannot access /dev/vx/dsk/olddg/vol01: A file or directory in
the path does not exist
# vi /etc/filesystems (NOTE: Edit the stanza for /testfs and have it point
to /dev/vx/dsk/newdg/vol01)
```

---

7. When removing a VERITAS volume containing a VERITAS file system, entries in /etc/filesystems must be manually edited and removed, or, alternatively, you can run the **rmfs** command, which will also remove the entry from the /etc/filesystems file, even for a VERITAS file system. This issue will be addressed in the next release.
8. When creating a new VERITAS File System, it is recommended not to use the **mkfs** command, because it does not update the /etc/filesystems file. Instead, use **crfs** or the VEA GUI, both of which will update the /etc/filesystems file. In the VEA GUI, you need to choose the **Add to file system table** option, which will update the /etc/filesystems file. See Figure 9-6 on page 297 for the VEA panel where you can specify this option. Start in the Volumes view, then choose **Actions -> File System -> New File System** to bring you to this panel.



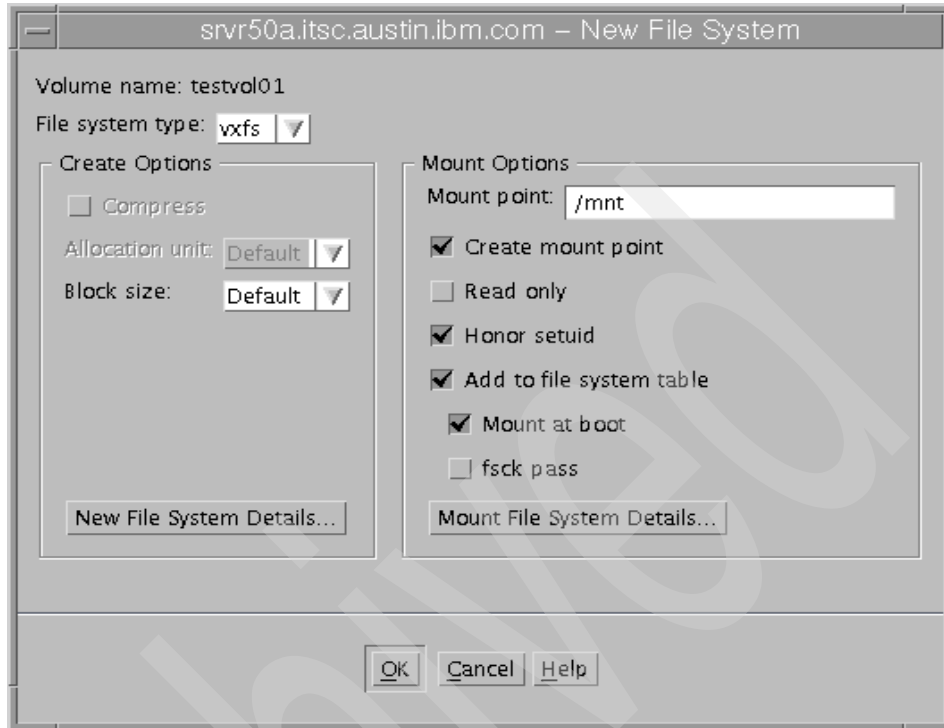


Figure 9-6 Create new file system with VEA GUI

9. When creating a volume and its layout using the VEA GUI, ensure that the `/etc/default/vxassist` file does not exist or is renamed. If the file exists, then the VEA GUI will use the volume layout information specified in that file instead of the information that you specify in VEA. If the file is not there or has been renamed, then VEA will create the volume and its layout as you specify in VEA.
10. The current release of VERITAS Volume Manager for AIX does not support root encapsulation or rootability.

## 9.5 References for troubleshooting

A *VERITAS Volume Manager 3.2 Troubleshooting Guide (AIX)* is available. You can obtain the guide in PDF format from the following Web site: <http://seer.support.veritas.com/docs/246779.htm>. The publication has in-depth information on what to do in case of any hardware failures, including information on recovering a mirrored volume, reattaching disks, and recovering a RAID 5 volume. Also included in this publication is information on error

messages that you might receive and how to resolve the errors. For example, kernel panic messages, **vxassist** error messages, and **vxconfigd** error messages are addressed, among others.

## LVM and VxVM command comparison tables

This appendix contains tables that compare commands and tasks between LVM and VxVM.

Both VERITAS Volume Manager and the Logical Volume Manager have a wealth of commands to manage volumes. For a complete command summary for VxVM, see the *VERITAS Volume Manager 3.2 - Administrator's Guide (AIX)*. For a complete understanding of LVM, see the AIX 5L Version 5.1 documentation available at:

[http://publibn.boulder.ibm.com/cgi-bin/ds\\_form?lang=en\\_US&viewset=AIX](http://publibn.boulder.ibm.com/cgi-bin/ds_form?lang=en_US&viewset=AIX)

There are also two Redbooks that cover the LVM in detail:

- ▶ *AIX Logical Volume Manager from A to Z: Introduction and Concepts*, SG24-5432
- ▶ *AIX Logical Volume Manager from A to Z: Troubleshooting and Commands*, SG24-5433

You can download both of these directly from IBM's Redbooks Web site at:

<http://www.redbooks.ibm.com>

Table A-1 shows a comparison of the commands that LVM and VxVM use to do common tasks. Most of these tasks can be accomplished from SMIT or smitty for both VxVM and LVM.

*Table A-1 Commands and tasks comparison in LVM and VxVM*

| <b>Task in LVM</b>  | <b>Equivalent task in VxVM</b>  | <b>AIX Logical Volume Manager commands</b>   | <b>VERITAS Volume Manager commands</b>  |
|---|---|--|---|
| Display information about a volume group.   | Display information about a disk group.   | <ul style="list-style-type: none"> <li>▶ <b>lsvg</b></li> <li>▶ <b>smitty lsvg</b></li> </ul>          | <ul style="list-style-type: none"> <li>▶ <b>vxprint -G</b></li> <li>▶ <b>vxdg</b></li> </ul>  |
| Create a volume group.  | Create a disk group.  | <ul style="list-style-type: none"> <li>▶ <b>mkvg</b></li> <li>▶ <b>smitty mkvg</b></li> </ul>          | <b>vxdg init</b>  |
| Change a volume group.  | Manage a disk group.  | <ul style="list-style-type: none"> <li>▶ <b>chvg</b></li> <li>▶ <b>smitty chvg</b></li> </ul>          | <ul style="list-style-type: none"> <li>▶ <b>vxdg</b></li> <li>▶ <b>smitty vxvmdg</b></li> </ul>   |
| Add a physical volume to a volume group.  | Add a physical disk to a disk group.  | <ul style="list-style-type: none"> <li>▶ <b>extendvg</b></li> <li>▶ <b>smitty extendvg</b></li> </ul>  | <ul style="list-style-type: none"> <li>▶ <b>vxdiskadd</b></li> <li>▶ <b>vxdg adddisk</b></li> <li>▶ <b>smitty vxvmdgsetdgadd</b></li> </ul>   |
| Remove a physical volume from a volume group.   | Remove a disk from a disk group.  | <ul style="list-style-type: none"> <li>▶ <b>reducevg</b></li> <li>▶ <b>smitty reducevg1</b></li> </ul> | <ul style="list-style-type: none"> <li>▶ <b>vxdiskadm</b></li> <li>▶ <b>vxdg rmdisk</b></li> <li>▶ <b>smitty vxvmdgsetdgremove</b></li> </ul> |
| Remove a volume group from the operating system.  | Remove a disk group from the operating system.  | <b>reducevg</b>  | <ul style="list-style-type: none"> <li>▶ <b>vxdg destroy</b></li> <li>▶ <b>smitty vxvmdiskadm</b></li> </ul>                                  |
| Export a volume group: Remove definition of a volume group from the operating system.                           | Disable access to a disk group.   | <b>exportvg</b>  | <ul style="list-style-type: none"> <li>▶ <b>vxdg deport</b></li> <li>▶ <b>smitty vxvmdgdeport</b></li> </ul>                                  |
| Rename a volume group on import (export first).   | Rename a disk group on import.  | <b>exportvg</b><br><b>importvg</b>   | <ul style="list-style-type: none"> <li>▶ <b>vxdg -tC -n</b></li> <li>▶ <b>smitty vxvmdgimport</b></li> </ul>                                  |
| Synchronize logical volumes, volume groups, or physical volumes. Activate volume group and its logical volumes. | Recover volumes on a disk, including plex attach, RAID 5 subdisk recovery, and resynchronizing volumes. | <b>syncvg</b><br><b>varyonvg</b>   | <b>vxrecover</b><br><b>vxmend fix clean</b>   |
| Mirror a volume group.  | Create a mirror.  | <ul style="list-style-type: none"> <li>▶ <b>mirrorvg</b></li> <li>▶ <b>smitty mirrorvg</b></li> </ul>  | <ul style="list-style-type: none"> <li>▶ <b>vxassist mirror</b></li> <li>▶ <b>smitty vxvmlvmirror</b></li> </ul>                              |
| Repair a mirror.  | Repair a mirror.  | <b>syncvg</b>  | <b>vxrecover</b>  |

| Task in LVM   | Equivalent task in VxVM   | AIX Logical Volume Manager commands   | VERITAS Volume Manager commands  |
|---|---|---|--|
| Unmirror a volume group.  | Disable a mirror.   | <code>unmirrorvg</code>   | <code>vxplex det</code>  |
| Display information about a logical volume.                                     | Display information about a volume.   | <code>lslv</code>   | <ul style="list-style-type: none"> <li>▶ <code>vxprint -v</code></li> <li>▶ <code>smitty vxvmlvlist</code></li> </ul>  |
| Create a logical volume in a volume group.                                      | Create a volume in a disk group.  | <ul style="list-style-type: none"> <li>▶ <code>mklv</code></li> <li>▶ <code>smitty mklv</code></li> </ul>         | <ul style="list-style-type: none"> <li>▶ <code>vxassist make</code></li> <li>▶ <code>smitty vxvmlvadd</code></li> </ul>  |
| Change characteristics of a logical volume.                                     | Change characteristics of a volume.   | <code>chlv</code>   | <ul style="list-style-type: none"> <li>▶ <code>vxedit set</code></li> <li>▶ <code>smitty vxvmlvset</code></li> </ul>   |
| Rename a logical volume.  | Rename a volume.  | <code>chlv</code>   | <ul style="list-style-type: none"> <li>▶ <code>vxedit -v rename name newname</code> (Note: Update the <code>/etc/filesystems</code> file with the new name)</li> <li>▶ <code>smitty vxvmlvrename</code></li> </ul> |
| Add a physical partition to a logical volume.                                   | Add a subdisk to a volume (Note: Increases a subdisk within a volume, if there is space). | <ul style="list-style-type: none"> <li>▶ <code>extendlv</code></li> <li>▶ <code>smitty extendlv</code></li> </ul> | <ul style="list-style-type: none"> <li>▶ <code>vxassist growto</code></li> <li>▶ <code>vxresize</code></li> <li>▶ <code>smitty vxvmlvgrow</code></li> </ul>  |
| Remove a logical volume from a volume group.                                    | Remove a volume from a disk group.  | <ul style="list-style-type: none"> <li>▶ <code>rmlv</code></li> <li>▶ <code>smitty rmlv</code></li> </ul>         | <ul style="list-style-type: none"> <li>▶ <code>vxedit rm</code></li> <li>▶ <code>smitty vxvmlvremove</code></li> </ul>   |
| Create a copy of a logical volume.  | Create a copy of a plex. Create a snapshot copy of a volume.                              | <code>cplv</code>   | <ul style="list-style-type: none"> <li>▶ <code>vxplex cp</code></li> <li>▶ <code>vxassist snapshot</code></li> <li>▶ <code>smitty vxvmlvsnapshot</code></li> </ul>   |
| Split copies from one logical volume and create a new logical volume from them. | Split a mirrored logical volume into two logical volumes.                                 | <code>splitlvcopy</code>  | <ul style="list-style-type: none"> <li>▶ <code>vxassist snapshot</code></li> <li>▶ <code>smitty vxvmlvsnapshot</code></li> </ul>   |
| Remove copies from a logical volume.  | Remove the snapshot copy.   | <code>rmlvcopy</code>   | <ul style="list-style-type: none"> <li>▶ <code>vxassist snapback</code></li> <li>▶ <code>smitty vxvmlvsnapback</code></li> </ul>   |
| Create a RAID layout for a logical volume.                                      | Create a RAID layout for a plex.  | <code>smitty lv</code>  | <ul style="list-style-type: none"> <li>▶ <code>vxassist make</code></li> <li>▶ <code>smitty vxvmlvadd</code></li> </ul>  |
| Create a mirror of a logical volume.  | Create a mirror of a plex.  | <code>mklv copy</code>  | <ul style="list-style-type: none"> <li>▶ <code>vxplex assoc</code></li> <li>▶ <code>vxassist mirror</code></li> </ul>  |

| <b>Task in LVM</b>  | <b>Equivalent task in VxVM</b>   | <b>AIX Logical Volume Manager commands</b>   | <b>VERITAS Volume Manager commands</b>  |
|---|--|--|---|
| Manage a logical volume.  | Manage a volume or a plex.   | <ul style="list-style-type: none"> <li>▶ <b>chlv</b></li> <li>▶ <b>smitty lv</b></li> </ul>                              | <ul style="list-style-type: none"> <li>▶ <b>vxvol</b></li> <li>▶ <b>smitty vxvmlv</b></li> <li>▶ <b>vxplex</b></li> </ul>   |
| Increase size of a logical volume.  | Increase size of a volume.   | <ul style="list-style-type: none"> <li>▶ <b>extendlv</b></li> <li>▶ <b>chlv</b></li> <li>▶ <b>smitty chlv</b></li> </ul> | <ul style="list-style-type: none"> <li>▶ <b>vxassist growto</b></li> <li>▶ <b>vxassist growby</b></li> <li>▶ <b>smitty vxvmlvgrow</b></li> </ul>                        |
| Display information about a physical volume (a physical disk).                | Display information about: <ul style="list-style-type: none"> <li>▶ A subdisk</li> <li>▶ A plex</li> <li>▶ A disk</li> </ul> | <ul style="list-style-type: none"> <li>▶ <b>lspv</b></li> <li>▶ <b>lsdev -C -c disk</b></li> </ul>                       | <ul style="list-style-type: none"> <li>▶ <b>vxprint -s</b></li> <li>▶ <b>vxprint -p</b></li> <li>▶ <b>vxdisk list</b></li> </ul>  |
| Administer disks.   | Administer disks.  | <b>smitty pv</b>   | <ul style="list-style-type: none"> <li>▶ <b>vxdiskadm</b></li> <li>▶ <b>smitty vxvmdiskadm</b></li> </ul>   |
| Set up a disk for use.  | Set up disk for use.   | <b>smitty pv</b>   | <ul style="list-style-type: none"> <li>▶ <b>vxdisksetup</b></li> <li>▶ <b>vxmake</b></li> <li>▶ <b>vxassist make</b></li> <li>▶ <b>smitty vxvmdiskadmadd</b></li> </ul> |
| Change characteristics of a physical volume.                                  | Manage a disk.   | <ul style="list-style-type: none"> <li>▶ <b>chpv</b></li> <li>▶ <b>smitty chpv</b></li> </ul>                            | <ul style="list-style-type: none"> <li>▶ <b>vxdiskadm</b></li> <li>▶ <b>vxdisk</b></li> </ul>   |
| Create a hot spare of a physical volume.                                      | Create a hot-relocation spare of a disk.   | <b>chpv -h</b>   | <b>vxedit set spare=on</b>  |
| Put a physical volume in unavailable state.                                   | Put a disk in the offline state.   | <b>chpv -v r</b>   | <ul style="list-style-type: none"> <li>▶ <b>vxdisk offline</b></li> <li>▶ <b>smitty vxvmdiskadmooffline</b></li> </ul>  |
| Put a physical volume in the available state.                                 | Put a disk in the online state.  | <b>chpv -v a</b>   | <ul style="list-style-type: none"> <li>▶ <b>vxdisk online</b></li> <li>▶ <b>smitty vxvmdiskadmonline</b></li> </ul>   |
| Move physical partitions from one physical volume to another physical volume. | Evacuate volumes from a disk (moves subdisks to another volume).   | <b>migratepv</b>   | <ul style="list-style-type: none"> <li>▶ <b>vxevac -g</b></li> <li>▶ <b>vxdiskadm</b></li> <li>▶ <b>vxassist move</b></li> </ul>  |
| Replace a physical volume with another physical volume.                       | Replace a disk with another disk.  | <b>replacepv</b>   | <b>vxdiskadm</b>  |

### Tasks with no direct LVM equivalents

Table A-2 lists VxVM tasks which have no direct LVM equivalent and also shows, in the third column, information about this task in LVM.

Table A-2 VxVM tasks with no LVM equivalent

| Task in VxVM  | VxVM command  | LVM information   |
|---|---|---|
| Rename a disk.  | <code>vxedit rename</code>  | Create disk name once at creation.                                |
| Add a DRL log to a volume.  | <code>vxassist addlog vol_name</code>   | Automatically handled by VGSA (Volume Group Status Area) logging. |
| Remove a DRL log from a volume.   | <code>vxassist remove log</code>  | Not needed in LVM. Handled automatically by VGSA.                 |
| Decrease disk space allocated to a logical volume with the <code>shrinkto</code> or <code>shrinkby</code> parameters. | <code>vxassist shrinkto</code><br>Note: Make sure you shrink the file system before shrinking the volume. | Not available in LVM at the time of writing this book.            |

### Tasks with no direct VxVM equivalent

Table A-3 shows a LVM task that has no direct VxVM equivalent and also shows, in the third column, information about this task in VxVM.

Table A-3 LVM tasks with no VxVM equivalent

| Task in LVM              | LVM command             | VxVM information   |
|--------------------------|-------------------------|--|
| Recreate a volume group. | <code>recreatevg</code> | Can do with several commands:<br><code>vxvg split</code><br><code>vxvg move</code> |





## JFS/JFS2 and VxFS command comparison tables

This appendix contains tables that compare commands and tasks between JFS or JFS2 and VxFS.

Both VxFS and the JFS or JFS2 have many commands to manage file systems. For a quick command summary for VxFS, see the *VERITAS File System 3.4 - Administrator's Guide (AIX)*. For an understanding of JFS and JFS2 commands, see the AIX 5L Version 5.1 documentation available at:

[http://publibn.boulder.ibm.com/cgi-bin/ds\\_form?lang=en\\_US&viewset=AIX](http://publibn.boulder.ibm.com/cgi-bin/ds_form?lang=en_US&viewset=AIX)

There are also two Redbooks that do cover some elements of the JFS and JFS2:

- ▶ *Problem Solving and Troubleshooting in AIX 5L*, SG24-5496
- ▶ *AIX 5L Differences Guide Version 5.1 Edition*, SG24-5765

These books can be downloaded directly from IBM's Redbooks Web site at:

<http://www.redbooks.ibm.com>

Table B-1 on page 306 shows a comparison of the commands that JFS/JFS2 and VxFS use to do these common file system tasks. Since many of the native file systems commands on AIX 5L simply use `/etc/filesystems`, some of the same

commands can be used for a VERITAS file system. Also, most of these tasks can be accomplished from SMIT or smitty for JFS, JFS2, and VxFS.

*Table B-1 Commands and tasks comparison in JFS/JFS2 and VxFS*

| <b>Task in JFS/JFS2</b>                                | <b>Equivalent task in VxFS</b>                         | <b>AIX JFS/JFS2 command</b>  | <b>VERITAS File System command</b>  |
|--|--|--|---|
| Display information about a file system.               | Display information about a file system.               | <b>lsfs</b>  | <b>lsfs</b>   |
| Make a file system.                                    | Create a file system.                                  | <ul style="list-style-type: none"> <li>▶ <b>crfs</b></li> <li>▶ <b>mkfs</b></li> </ul>   | <b>crfs</b>   |
| Increase a file system size.                           | Extend a file system size.                             | <b>chfs</b>  | <b>fsadm</b>  |
| Create a snapshot of a file system for online backups. | Create a snapshot of a file system for online backups. | <b>chfs -a splitcopy</b>   | <b>mount -V vxfs -o snapof</b>  |
| Remove a file system.                                  | Remove a file system.                                  | <b>rmfs</b>  | <b>rmfs</b>   |
| Increase a file system's contiguous free space.        | Reorganize (or compact) a fragmented file system.      | <ul style="list-style-type: none"> <li>▶ <b>defragfs</b></li> <li>▶ <b>smitty dejfs</b></li> <li>▶ <b>smitty dejfs2</b></li> </ul>       | <b>fsadm</b>  |
| Mount a file system.                                   | Mount a file system.                                   | <ul style="list-style-type: none"> <li>▶ <b>mount -V jfs</b></li> <li>▶ <b>mount -V jfs2</b></li> <li>▶ <b>smitty mountfs</b></li> </ul> | <ul style="list-style-type: none"> <li>▶ <b>mount -V vxfs</b></li> <li>▶ <b>smitty mountfs</b></li> </ul> |
| Unmount a file system.                                 | Unmount a file system.                                 | <ul style="list-style-type: none"> <li>▶ <b>umount</b></li> <li>▶ <b>smitty umountfs</b></li> </ul>                                      | <ul style="list-style-type: none"> <li>▶ <b>umount</b></li> <li>▶ <b>smitty umountfs</b></li> </ul>       |
| Back up a file system                                  | Back up a file system                                  | <ul style="list-style-type: none"> <li>▶ <b>backup</b></li> <li>▶ <b>smitty backfilesys</b></li> </ul>                                   | <ul style="list-style-type: none"> <li>▶ <b>vxdump</b></li> <li>▶ <b>smitty backfilesys</b></li> </ul>    |
| Restore a file system.                                 | Restore a file system.                                 | <ul style="list-style-type: none"> <li>▶ <b>restore</b></li> <li>▶ <b>smitty restfilesys</b></li> </ul>                                  | <ul style="list-style-type: none"> <li>▶ <b>vxrestore</b></li> <li>▶ <b>smitty restfilesys</b></li> </ul> |
| Create an online backup of a file system.              | Create an online backup of a file system.              | <b>chfs -a splitcopy=/backup backup restore rmfs /backup</b>   | <b>mount -V vxfs -o snapof vxdump vxrestore</b>   |

**Tasks with no direct LVM equivalents**

Table B-2 lists a task that has no direct JFS/JFS2 equivalent.

*Table B-2 VxFS task with no JFS/JFS2 equivalent*

| Task in VxFS          | VxFS command       | JFS/JFS2 information  |
|-----------------------|--------------------|---|
| Shrink a file system. | <code>fsadm</code> | Not available for JFS or JFS2 at the time of writing this book. |

**Tasks with no direct VxVM equivalents**

Table B-3 lists a task that has no direct VxFS equivalent.

*Table B-3 JFS/JFS2 task with no VxFS equivalent*

| Task in JFS/JFS2              | JFS/JFS2 command     | VxFS information                               |
|-------------------------------|----------------------|--|
| Make a prototype file system. | <code>mkproto</code> | No equivalence in VxFS, but not needed in VxFS |



## Sample installation scripts

This appendix contains two sample installation scripts that can be used for carrying out multiple installs to new hosts. One script is intended for installing to a single host, while the other can be used for installing to multiple hosts, either simultaneously or sequentially, by calling the other script. The scripts are:

- ▶ `Install_All_Hosts.ksh`
- ▶ `Install_Host.ksh`

These scripts have been tested in a lab with two F50s and an IBM @server pSeries p610. One F50 was used as the install server and the other two were used for base installs. Of the two systems being installed, the IBM @server pSeries p610 contained three internal disks of 18 GB while the other F50 contained five internal disks of 9 GB. These scripts are not the product of VERITAS or IBM development, but are merely an example of how an automated installation can be achieved with minimal input on multiple systems.

**Attention:** These scripts are not supported by VERITAS or IBM; please do not log any support calls in the event of problems using these scripts. The scripts have been tested in a lab environment. Customizations may be made to these scripts to suite any environment as needed.

To set up an install server, perform the following steps:

1. Mount the installation CD on the install server or copy the \*.bff files from the installation CD to a directory on the server.
2. Export the file system containing the install files to all the hosts that the install will take place on.
3. On the hosts to be installed, set up a .rhosts file to enable the install server access to the hosts. These can be removed after the install.
4. Copy the scripts Install\_All\_Hosts.ksh and Install\_Host.ksh to a directory on the install server. These scripts can be seen in Example C-1 and Example C-2 on page 315.
5. Edit the Install\_All\_Hosts.ksh and change the HOME\_DIR variable to the path name that the Install\_All\_Hosts.ksh and Install\_Host.ksh scripts were copied in to.
6. Create a file in the directory that the scripts were copied in to called VRTSinstall\_config containing the configuration information. See "Configuration file options" on page 318 for information on how to set up this file.

*Example: C-1 Install\_All\_Hosts.ksh*

---

```
#!/bin/ksh

# Script : Install_All_Hosts.ksh
#
# Purpose : To Install VxFS & VxVM on Multiple hosts
#           via the Install_Host.ksh script. This script
#           reads configuration information from a file
#           specified in the $CONFIG_FILE variable.

HOME_DIR=/usr/local/bin/VFSinstall
CONFIG_FILE=$HOME_DIR/VRTSinstall_config
INSTALL_SERVER=`uname -n`
BASE_DIR=`grep "^BASE_DIR=" $CONFIG_FILE | sed "s/BASE_DIR=\\///g"`
HOST_LIST=`awk -F= '$1=="HOST" {print $2}' $CONFIG_FILE`
INSTALL_MODE=`awk -F= '$1=="INSTALL_MODE" {print toupper($2)}' $CONFIG_FILE`
LOGFILE=$HOME_DIR/Install.log
PKGS_DIR=`awk -F= '$1=="PKGS_DIR" {print $2}' $CONFIG_FILE`

Check_Exports()
{
    #--- Check Packages directory exists and contains the packages

    if [ ! -d "$PKGS_DIR" ]
    then
        echo "$PKGS_DIR does not exist - Install cannot continue"
```

```

        exit 1
    fi

    if [ `ls $PKGS_DIR/VRTS*bff | wc -l` -le 10 ]
    then
        echo "Not all packages for VxFS & VxVM"
        echo "seem to be in $PKGS_DIR"
        echo "Install cannot continue"
        exit 1
    fi

    #--- Check $BASE_DIR is exported from $INSTALL_SERVER
    showmount -e | grep -q "$BASE_DIR"
    if [ $? -ne 0 ]
    then
        echo "$BASE_DIR not exported - cannot continue"
        exit 1
    fi

    #--- Check that $BASE_DIR is exported to all hosts in $CONFIG_FILE
    for HOST in `echo $HOST_LIST`
    do
        showmount -e | grep "$BASE_DIR" | grep -q $HOST
        if [ $? -ne 0 ]
        then
            echo "$BASE_DIR not exported to $HOST - cannot continue"
            echo "Please check $CONFIG_FILE and re run this script."
            exit 1
        else
            echo "$BASE_DIR exported to $HOST "
        fi
    done

    #--- Check rsh to hosts to install
    for HOST in `echo $HOST_LIST`
    do
        rsh $HOST ls -la >/dev/null 2>&1
        if [ $? -ne 0 ]
        then
            echo "Cannot rsh to $HOST - install cannot continue"
            echo "Please check .rhosts file on $HOST and re run this script."
            exit 1
        else
            echo "rsh to $HOST - ok"
        fi
    done
done
}

Check_Config()

```

```

{
#--- Check the disks on the remote system are OK for installation if
#   INIT_VXVM is set to Y or y.
for HOST in `echo $HOST_LIST`
do
    grep -p "^HOST=${HOST}" $CONFIG_FILE > /tmp/${HOST}_config
    rsh $HOST lspv > /tmp/${HOST}_lspv.txt
    INIT_VXVM=`awk -F= ' $1=="INIT_VXVM" {print toupper($2)}'
/tmp/${HOST}_config`
    if [ "$INIT_VXVM" = "Y" ]
    then
        for DISK in `grep "^DISKS=" /tmp/${HOST}_config | sed "s/DISKS=//g"`
        do
            STATUS=`grep $DISK /tmp/${HOST}_lspv.txt | awk '{print $3}'`
            if [ "$STATUS" != "None" ]
            then
                echo "$DISK on $HOST seems to be already in use"
                echo "Please check - Install cannot continue"
                exit 1
            fi
        done

#--- Check there is an accessname for each disk to be added.
DISKS=`grep "^DISKS=" /tmp/${HOST}_config | sed "s/DISKS=//g" | wc -w`
NAMES=`grep "^NAMES=" /tmp/${HOST}_config | sed "s/DISKS=//g" | wc -w`
if [ $DISKS -ne $NAMES ]
then
    echo "Not all access names specified for DISKS on $HOST"
    echo "Please check $CONFIG_FILE as install cannot continue"
    exit 1
fi
fi

#--- Check mount point exists on Host.
MOUNT_POINT=`grep "^MOUNT_POINT=" /tmp/${HOST}_config | sed
"s/MOUNT_POINT=\\//g"`
rsh $HOST "ls -la /${MOUNT_POINT}" >/dev/null 2>&1
if [ $? -ne 0 ]
then
    echo "Mount point does not exist on $HOST"
    echo "Install cannot continue"
    exit 1
fi

#--- Build the remote path to installation files.
SUB_PATH=`echo $PKGS_DIR | tr "/" ":"`
HOST_MNT_PNT=`awk -F= ' $1=="MOUNT_POINT" {print $2}' /tmp/${HOST}_config
| tr -d "/"`
HOST_INST_PATH=`echo $SUB_PATH | sed "s/$BASE_DIR/$HOST_MNT_PNT/g" | tr
":" "/"`

```



```

        echo "INSTALL_PATH=${HOST_INST_PATH}" >> /tmp/${HOST}_config

        rcp /tmp/${HOST}_config ${HOST}:/tmp/${HOST}_config
    done
}

Build_Script()
{
    #--- Change the variables in the host install script and
    #    copy to host.
    for HOST in `echo $HOST_LIST`
    do
        sed "s/installservertogoher/${INSTALL_SERVER}/g"
$HOME_DIR/Install_Host.ksh > $HOME_DIR/Install_${HOST}.1
        sed "s/mountpointtogoher/${MOUNT_POINT}/g" $HOME_DIR/Install_${HOST}.1 >
$HOME_DIR/Install_${HOST}.2
        sed "s/basedirtogoher/${BASE_DIR}/g" $HOME_DIR/Install_${HOST}.2 >
$HOME_DIR/Install_${HOST}
        rm $HOME_DIR/Install_${HOST}.*
        rcp $HOME_DIR/Install_${HOST} ${HOST}:/tmp
        rsh $HOST "chmod 700 /tmp/Install_${HOST}"
    done
}

Install_Parallel()
{
    #--- Install on hosts simultaneously
    echo "**** Installing on following hosts: \n$HOST_LIST"
    for HOST in `echo $HOST_LIST`
    do
        rsh $HOST -n "echo /tmp/Install_${HOST} | at now >/dev/null 2>&1"
    done
    Check_Finished

    [[ -f "$LOGFILE" ]] && rm $LOGFILE
    for HOST in `echo $HOST_LIST`
    do
        echo "*** ${HOST} ****" >> $LOGFILE
        rsh $HOST "cat /tmp/${HOST}.status" >> $LOGFILE
    done
    echo "\n***Installation complete "
    echo "A log of the install can be seen by"
    echo "viewing $LOGFILE"
}

Install_Seq()
{
    #--- Install on hosts sequentially
    [[ -f "$LOGFILE" ]] && rm $LOGFILE

```

```

echo "\n*** The following Hosts will be installed:\n$HOST_LIST"
for HOST in `echo $HOST_LIST`
do
    echo "\nInstalling on Host : $HOST"
    rsh $HOST -n "echo /tmp/Install_${HOST} | at now >/dev/null 2>&1"
    INST_RUNNING=`rsh $HOST "ps -ef | grep Install_${HOST} | grep -v grep"`
    while [ -n "$INST_RUNNING" ]
    do
        sleep 120
        INST_RUNNING=`rsh $HOST "ps -ef | grep Install_${HOST} | grep -v grep"`
    done
    echo "*** ${HOST} ****" >> $LOGFILE
    rsh $HOST "cat /tmp/${HOST}.status" >> $LOGFILE
    echo "$HOST installation complete"
done
echo "\n***Installation complete "
echo "A log of the install can be seen by"
echo "viewing $LOGFILE"
}

Check_Finished()
{
    #--- Check the installation of all hosts has completed if
    #    installing in parallel.
    FINISHED="N"
    HOSTS_STILL_INST=""
    while [ "$FINISHED" != "Y" ]
    do
        sleep 120
        for HOST in `echo $HOST_LIST`
        do
            INST_RUNNING=`rsh $HOST "ps -ef | grep Install_${HOST} | grep -v grep"`
            if [ -n "$INST_RUNNING" ]
            then
                HOSTS_STILL_INST="$HOSTS_STILL_INST $HOST"
            fi
        done
        if [ -n "$HOSTS_STILL_INST" ]
        then
            echo "*** Waiting for Hosts: $HOSTS_STILL_INST"
            HOSTS_STILL_INST=""
        else
            FINISHED="Y"
        fi
    done
}

```

```
#--- Main. -----
if [ -f "$CONFIG_FILE" ]
then
    Check_Exports
    Check_Config
    Build_Script
    case "$INSTALL_MODE" in
        P) Install_Parallel ;;
        S) Install_Seq ;;
        *) echo "Incorrect Mode specified in $CONFIG File - must be P or S" ;;
    esac
else
    echo "No configuration file found - cannot install"
fi
```

---

#### *Example: C-2 Install\_Host.ksh*

---

```
#!/bin/ksh

# Script: Install_Host.ksh
#
# Purpose: To install VxFS and VxVM from a remote server. This
#          script can be run on its own or as part of a multiple
#          install from Install_All_Hosts.ksh. If running on its own
#          change the INSTALL_HOST, BASE_DIR and MNT_PNT variables to
#          the host that contains the install files, the exported mount
#          point from the host with the install files and mount point to
#          the local NFS mount point where the remote servers directory
#          will be mounted. Also the Configuration file will need to be
#          manually created.

#---- Variables -----
INSTALL_HOST="installservertogoehere"
CONFIG_FILE="/tmp/~uname -n~_config"
LOGFILE="/tmp/VFS_install.log"
BASE_DIR="/basedirtogoehere"
MNT_PNT="/mountpointtogoehere"
BIN_DIR="/usr/sbin"
STATUS_FILE="/tmp/~uname -n~.status"
PKGS_DIR=`awk -F= ' $1=="INSTALL_PATH" {print $2}' $CONFIG_FILE`

#---- Functions -----
INSTALL_FILESETS()
{
#--- Install all filesets required for
#--- VERITAS Foundation Suite for AIX.
#--- Install license software, VxFS & VxVM filesets
    echo "\n**** Installing VxFS **** \n" > $LOGFILE
    installp -acXd $PKGS_DIR VRTSvlic VRTSvxfs \
```

```

VRTSfsdoc >>$LOGFILE 2>&1
echo "\n**** Installing VxVM *****\n" >> $LOGFILE
installp -acXd $PKGS_DIR VRTSvxvm VRTSvmman VRTSvmdoc \
>>$LOGFILE 2>&1
echo "\n**** Installing VEA and Provider Packages ****\n" >> $LOGFILE 2>&1
installp -acXd $PKGS_DIR VRTSob VRTSobgui VRTSfspro \
VRTSvmpro >> $LOGFILE 2>&1
grep -qi failed $LOGFILE
if [ $? -eq 0 ]
then
    echo "\tErrors found in installp process" >> $STATUS_FILE
    echo "\tInstall cannot continue" >> $STATUS_FILE
    exit 1
else
    echo "\tAll filesets installed successfully on `uname -n`" >>
$STATUS_FILE
fi
}

INSTALL_LICENSE()
{
#---- Install VxFS & VxVM licenses.
VXFS_LIC=`awk -F= ' $1=="VRTSvxfs_lic" {print $2}' $CONFIG_FILE`
VXVM_LIC=`awk -F= ' $1=="VRTSvxvm_lic" {print $2}' $CONFIG_FILE`
echo "\nInstalling VxFS License key \n" >> $LOGFILE
echo $VXFS_LIC | /sbin/vxlicinst >> $LOGFILE 2>&1
echo "\nInstalling VxVM License key \n" >> $LOGFILE
echo $VXVM_LIC | /sbin/vxlicinst >> $LOGFILE 2>&1

#--- Check licenses have installed corectly.
#--- Install can continue if VxFS license is not loaded
#--- but not for VxVM.
vxlictest -n "VERITAS File System" -f "VXFS" -sup
if [ $? -eq 0 ]
then
    echo "\tVxFS license sucessfully installed on `uname -n`" >>
$STATUS_FILE
else
    echo "\tVxFS license unsuccessfully loaded on `uname -n`" >>
$STATUS_FILE
fi
vxlictest -n "VERITAS Volume Manager" -f "VxVM" -sup
if [ $? -eq 0 ]
then
    echo "\tVxVM license sucessfully installed on `uname -n`" >>
$STATUS_FILE
else
    echo "\tVxVM license unsuccessfully loaded on `uname -n`" >> $STATUS_FILE
    echo "\tInstall cannot continue" >> $STATUS_FILE

```

```

        exit 1
    fi
}

INIT_VXVM()
{
    #--- Initialize VxVM using disks listed in config file.
    INIT_VRTS=`awk -F= '$1=="INIT_VXVM" {print toupper($2)}' $CONFIG_FILE`
    if [ "$INIT_VRTS" = "Y" ]
    then
        DISKS=`awk -F= '$1=="DISKS" {print $2}' $CONFIG_FILE`
        set -A NAMES `awk -F= '$1=="NAMES" {print $2}' $CONFIG_FILE`
        ${BIN_DIR}/vxconfigd -m disable
        ${BIN_DIR}/vxdctl init
        ${BIN_DIR}/vxdg init rootdg
        NUM=0
        for DISK in `echo $DISKS`
        do
            ACC_NAME=`echo ${NAMES[$NUM]}`
            /usr/sbin/chpv -C $DISK
            ${BIN_DIR}/vxdisk -f init $DISK
            ${BIN_DIR}/vxdg -g rootdg adddisk ${ACC_NAME}=${DISK}
            (( NUM = $NUM + 1 ))
        done
        ${BIN_DIR}/vxconfigd -k
        ${BIN_DIR}/vxdctl enable
        ${BIN_DIR}/vxiod set 10
        #--- Remove next file so startup scripts will work.
        rm /etc/vx/reconfig.d/state.d/install-db
        echo "\tVxVM initialization complete on `uname -n`" >> $STATUS_FILE
    else
        echo "\tVxVM initialization flag not set -" >> $STATUS_FILE
        echo "\tneed to run vxinstall manually on `uname -n`" >> $STATUS_FILE
    fi
}

#--- Main -----
[[ -f "$LOGFILE" ]] && rm $LOGFILE
[[ -f "$STATUS_FILE" ]] && rm $STATUS_FILE
mount -o ro ${INSTALL_HOST}:${BASE_DIR} ${MNT_PNT}
if [ $? -ne 0 ]
then
    echo "NFS mount Failed on `uname -n`" >> $STATUS_FILE
else
    INSTALL_FILESETS
    INSTALL_LICENSE
    INIT_VXVM
fi
umount ${MNT_PNT}

```

## Configuration file options

The configuration file is divided into two sections: the global section and the hosts section. Each variable placed in this file takes the form of Variable=Value. Comments can be placed in this file as long as they are preceded with a “#” character and are not on a line containing a variable. Example C-3 shows the layout of the configuration file.

### *Example: C-3 Configuration file*

---

```
# This file is read by Install_All_Hosts.ksh and contains variables
# for installing to multiple hosts.
```

```
#--- Global Section
```

```
BASE_DIR=/cdrom
```

```
INSTALL_MODE=S
```

```
PKGS_DIR=/cdrom/other/VRTSFST/pkgs
```

```
#--- Hosts to install. Separate each host by a blank line.
```

```
HOST=srvr50a
```

```
MOUNT_POINT=/mnt
```

```
VRTSvxfs_lic=63969938967390075648898
```

```
VRTSvxvm_lic=59689032508420444016955
```

```
INIT_VXVM=Y
```

```
DISKS=hdisk1 hdisk2
```

```
NAMES=diska diskb
```

```
HOST=srvr60a
```

```
MOUNT_POINT=/mnt
```

```
VRTSvxfs_lic=63069613697805629648898
```

```
VRTSvxvm_lic=59642249071562707016955
```

```
INIT_VXVM=N
```

```
DISKS=hdisk1 hdisk2
```

```
NAMES=disk01 disk02
```

---

The global section has the following options:

- ▶ **BASE\_DIR:** This is the file system containing the install files that is exported from the install server and remote mounted on the hosts to be installed.
- ▶ **INSTALL\_MODE:** This determines the method of install and has two options: P for parallel and S for sequential. In parallel mode, the hosts to be installed will be installed simultaneously. In sequential mode, the hosts to be installed will be installed one at a time.

- ▶ **PKGS\_DIR:** This is the full path name to the location of the install files on the server.

The hosts section contains options for each host. Each host in the configuration file should be separated by a blank line. The options are:

- ▶ **HOST:** This is the host name of the client to be installed on.
- ▶ **MOUNT\_POINT:** This is the mount point on the host where the file system on the install server will be NFS mounted to on the client to be installed.
- ▶ **VRTSvxfs\_lic:** This is the license key required for VERITAS File System. The license key must be valid.
- ▶ **VRTSvxvm\_lic:** This is the license key required for VERITAS Volume Manager. The license key must be valid.
- ▶ **INIT\_VXVM:** This determines whether or not to initialize VxVM on the client after the packages have been installed. If set to Y, the install script will start up the volume manager processes and create a rootdg with the disks specified. Any other option other than Y or y, if placed here, will not initialize VxVM and **vxinstall** will have to be manually run on the client. This may be useful if multipathing options need to be changed, as this script does not do any multipathing changes.
- ▶ **DISKS:** This is a list of disks that will be added to the rootdg on the client when VxVM is initialized. At least one disk must be placed here if INIT\_VXVM is set to Y or y.
- ▶ **NAMES:** This is the name VxVM will give to the disks that are added to the rootdg and must correspond to the names listed in the DISKS parameter if the INIT\_VXVM parameter is set to Y or y.

### Installing on one host from an NFS mounted directory

The **Install\_Host.ksh** script can be used to install just one host. This could be used as part of a NIM install process. To use **Install\_Host.ksh** to install on a single host, some modifications need to be made to the script as follows:

- ▶ Change the **INSTALL\_HOST** variable to the host name of the install server.
- ▶ Change the **BASE\_DIR** variable to the file system that contains the install files that is to be NFS mounted from the install server.
- ▶ Change the **MNT\_PNT** variable to the path where the local NFS mount point will be.

Also, a configuration file needs to be created in the **/tmp** directory called **<hostname>\_config**, where **<hostname>** is the output from the **uname -n** command. For example, to install on a host called **serverb**, the configuration file

name will be called /tmp/serverb\_config. The configuration file needs to contain the following variables:

- ▶ VRTSvxfs\_lic
- ▶ VRTSvxvm\_lic
- ▶ INIT\_VXVM
- ▶ DISKS
- ▶ NAMES

For descriptions of these variables, see “Configuration file options” on page 318. In addition to the above variables, the following line in Install\_Host.ksh needs to be changed:

```
PKGS_DIR=`awk -F= ' $1=="INSTALL_PATH" {print $2}' $CONFIG_FILE`
```

Change this variable to the full path name of the install files on the client after the remote file system has been mounted. For example, if the path to the install files is /mnt/images/pkgs, the variable will be:

```
PKGS_DIR="/mnt/images/pkgs"
```

## Installing on one host from a local CD-ROM or local file system

If the host being installed contains the install files either on a CD-ROM or a file system containing the installation .bff files, then the following items need to be changed in the Install\_Host.ksh script:

- ▶ Remove all lines from the main section of the script apart from the following five lines:

```
[[ -f "$LOGFILE" ]] && rm $LOGFILE
[[ -f "$STATUS_FILE" ]] && rm $STATUS_FILE
INSTALL_FILESETS
INSTALL_LICENSE
INIT_VXVM
```

- ▶ Change the PKGS\_DIR variable to the full path name of the directory containing the install files. For example, if installing from the AIX bonus pack CD, the variable will be:

```
PKGS_DIR="/cdrom/other/VRTSFST/pkgs"
```

- ▶ Create a configuration file called /tmp/<hostname>\_config, where <hostname> is the output from the **uname -n** command. The configuration file needs to contain VRTSvxfs\_lic, VRTSvxvm\_lic, INIT\_VXVM, DISKS, and NAMES.

## Running the scripts

Once the scripts are set up, an install can be run by either running **Install\_A11\_Hosts.ksh** from the install server for multiple system installs or by



running **Install\_Host.ksh** from a host for a single system install. Once the install is complete, a summary log file can be found in `$HOME_DIR/Install.log`.

**Important:** Before running this install, ensure that all target disks on the systems to be installed to have a status of “None” when running the **lspv** command. If they are not, run the **chpv -C** command for each disk. The script will not initialize any disk that is already under control of either VxVM or LVM.



# The VERITAS Cluster Server for AIX

This appendix introduces VERITAS Cluster Server for AIX, which is a high availability software package that is designed to reduce both planned and unplanned downtime in a business critical environment.

Topics discussed include:

- ▶ Executive overview
- ▶ Components of a VERITAS cluster
- ▶ Cluster resources
- ▶ Cluster configurations
- ▶ Cluster communications
- ▶ Cluster installation and setup
- ▶ Cluster administration facilities
- ▶ HACMP and VERITAS Cluster Server compared

## Executive overview

VERITAS Cluster Server is a leading open systems clustering solution on Sun Solaris and is also available on HP/UX, AIX, Linux and Windows NT 4. It is scalable up to 16 nodes in an AIX cluster, and supports the management of multiple VCS clusters (NT or UNIX) from a single Web or Java based Graphical User Interface (GUI). However, individual clusters must be comprised of systems running the same operating system.

VERITAS Cluster Server has similar function to IBM High Availability Cluster Multi Processing (HACMP) product, eliminating single points of failure through the provision of redundant components, automatic detection of application, adapter, network, and node failures, and managing failover to a remote server with no apparent outage to the end user.

The VCS GUI based cluster management console provides a common administrative interface in a cross platform environment. There is also integration with other VERITAS products, such the VERITAS Volume Replicator and VERITAS Global Cluster Server.

## Components of a VERITAS cluster

A VERITAS cluster is comprised nodes, external shared disk, networks, applications, and clients. Specifically a cluster is defined as all servers with the same cluster ID connected via a set of redundant heartbeat paths.

- ▶ **Nodes:** Nodes in a VERITAS cluster are called cluster servers. There can be up to 16 cluster servers in an AIX VERITAS cluster, and up to 32 nodes on other platforms. A node will run an application or multiple applications, and can be added to or removed from a cluster dynamically.
- ▶ **Shared external disk devices:** VERITAS Cluster Server supports a number of third-party storage vendors, and works in small computer system interface (SCSI), network storage attached (NAS), and storage area networks (SAN) environments. In addition, VERITAS offer a Cluster Server Storage Certification Suite (SCS) for OEM disk vendors to certify their disks for use with VCS. Contact VERITAS directly for more information about SCS.
- ▶ **Networks and disk channels:** These channels, in VCS cluster networks, are required for both heartbeat communication, to determine the status of resources in the cluster, and also for client traffic. VCS uses its own protocol, Low Latency Transport (LLT), for cluster heartbeat communication. A second protocol, Group Membership Services/Atomic Broadcast (GAB), is used for communicating cluster configuration and state information between servers in the cluster. The LLT and GAB protocols are used instead of a TCP/IP based

communication mechanism. VCS requires a minimum of two dedicated private heartbeat connections, or high-priority network links, for cluster communication. To enable active takeover of resources, should one of these heartbeat paths fail, a third dedicated heartbeat connection is required.

Client traffic is sent and received over public networks. This public network can also be defined as a low-priority network, so should there be a failure of the dedicated high-priority networks, heartbeats can be sent at a slower rate over this secondary network. A further means of supporting heartbeat traffic is via disk, using what is called a GABdisk. Heartbeats are written to and read from a specific area of a disk by cluster servers. Disk channels can only be used for cluster membership communication, not for passing information about a cluster's state. Note that the use of a GABdisk limits the number of servers in a cluster to eight, and not all vendors disk arrays support GABdisks.

Ethernet is the only supported network type for VCS.

## Cluster resources

Resources to be made highly available include network adapters, shared storage, IP addresses, applications, and processes. Resources have a type associated with them and you can have multiple instances of a resource type. Control of each resource type involves bringing the resource online, taking it offline, and monitoring its health.

- ▶ Agents: For each resource type, VCS has a cluster agent that controls the resource. Types of VCS agents include:
  - *Bundled agents* are standard agents that come bundled with the VCS software for basic resource types, such as disk, IP, and mount. Examples of actual agents are *Application*, *IP*, *DiskGroup*, and *Mount*. For additional information, see the *VERITAS Bundled Agents Reference Guide*.
  - *Enterprise agents* are for applications, and are purchased separately from VCS. Enterprise agents exist for products such as DB2, Oracle, and VERITAS Netbackup.
  - *Storage agents* also exist to provide access and control over storage components, such as the VERITAS ServPoint (NAS) appliance.
  - *Custom agents* can be created using the VERITAS developer agent for additional resource types, including applications for which there is no enterprise agent. See the *VERITAS Cluster Server Agents Developers Guide* for information about creating new cluster agents.

VERITAS cluster agents are multithreaded, so they support the monitoring of multiple instances of a resource type.

- ▶ Resource categories: A resource also has a category associated with it that determines how VCS handles the resource. Resources categories include:
  - On-Off: VCS starts and stops the resource as required (most resources are On-Off).
  - On-Only: Brought online by VCS, but is not stopped when the related service group is taken offline. An example of this kind of resource would be starting a daemon.
  - Persistent: VCS cannot take the resource online or offline, but needs to use it, so it monitors its availability. An example would be the network card that an IP address is configured upon.
- ▶ Service group: A set of resources that are logically grouped to provide a service. Individual resource dependencies must be explicitly defined when the service group is created to determine the order resources are brought online and taken offline. When VERITAS cluster server is started, the cluster server engine examines resource dependencies and starts all the required agents. A cluster server can support multiple service groups.

Operations are performed on resources and also on service groups. All resources that comprise a service group will move if any resource in the service group needs to move in response to a failure. However, where there are multiple service groups running on a cluster server, only the affected service group is moved.

The service group type defines takeover relationships, which is either:

- Failover: The service group runs only one cluster server at a time and supports failover of resources between cluster server nodes. Failover can be both unplanned (unexpected resource outage) and planned, for example, for maintenance purposes. Although the nodes, which can take over a service group, will be defined, there are three methods by which the destination fail over node is decided:
  - Priority: The *SystemList* attribute is used to set the priority for a cluster server. The server with the lowest defined priority that is in the running state becomes the target system. Priority is determined by the order the servers are defined in the *SystemList* with the first server in the list being the lowest priority server. This is the default method of determining the target node at failover, although priority can also be set explicitly.
  - Round: The system running the smallest number of service groups becomes the target.

- Load: The cluster server with the most available capacity becomes the target node. To determine available capacity, each service group is assigned a capacity. This value is used in the calculation to determine the fail-over node, based on the service groups active on the node.
- Parallel: Service groups are active on all cluster nodes that run resources simultaneously. Applications must be able to run on multiple servers simultaneously with no data corruption. This type of service group is sometimes also described as concurrent. A parallel resource group is used for things like Web hosting.

The Web VCS interface is typically defined as a service group and kept highly available. It should be noted, however, that although actions can be initiated from the browser, it is not possible to add or remove elements from the configuration via the browser. The Java VCS console should be used for making configuration changes.

In addition, service group dependencies can be defined. Service group dependencies apply when a resource is brought online, when a resource faults, and when the service group is taken offline. Service group dependencies are defined in terms of a parent and child, and a service group can be both a child and parent. Service group dependencies are defined by three parameters:

- Category
- Location
- Type

Values for these parameters are:

- online/offline
- local/global/remote
- soft/hard

As an example, take two service groups with a dependency of online, remote, and soft. The category online means that the parent service group must wait for the child service group to be brought on online before it is started. Use of the remote location parameter requires that the parent and child must necessarily be on different servers. Finally, the type soft has implications for service group behavior should a resource fault. See the *VERITAS Cluster Server User Guide* for detailed descriptions of each option. Configuring service group dependencies adds complexity, so must be carefully planned.

- Attributes: All VCS components have attributes associated with them that are used to define their configuration. Each attribute has a *data type* and *dimension*. Definitions for data types and dimensions are detailed in the *VERITAS Cluster Server User Guide*. An example of a resource attribute is the IP address associated with a network interface card.

- ▶ System zones: VCS supports system zones, which are a subset of systems for a service group to use at initial failover. The service group will choose a host within its system zone before choosing any other host.

## Cluster configurations

The VERITAS terminology used to describe supported cluster configurations is:

- ▶ Asymmetric: There is a defined primary and a dedicated backup server. Only the primary server is running a production workload.
- ▶ Symmetric: There is a two node cluster where each cluster server is configured to provide a highly available service and acts as a backup to the other.
- ▶ N-to-1: There are N production cluster servers and a single backup server. This setup relies on the concept that failure of multiple servers at any one time is relatively unlikely. In addition, the number of slots in a server limits the total number of nodes capable of being connected in this cluster configuration.
- ▶ N+1: An extra cluster server is included as a spare. Should any of the N production servers fail, its service groups will move to the spare cluster server. When the failed server is recovered, it simply joins as a spare so there is no further interruption to service to failback the service group.
- ▶ N-to-N: There are multiple service groups running on multiple servers, which can be failed to potentially different servers.

## Cluster communication

Cross cluster communication is required to achieve automated failure detection and recovery in a high availability environment. Essentially all cluster servers in a VERITAS cluster must run:

- ▶ High availability daemon (HAD): This is the primary process and is sometimes referred to as the cluster server engine. A further process, *hashadow*, monitors HAD and can restart it if required. VCS agents monitor the state of resources and pass information to their local HAD. The HAD then communicates information about cluster status to the other HAD processes using the GAB and LLT protocols.
- ▶ Group membership services/atomic broadcast (GAB): GAB operates in the kernel space, monitors cluster membership, tracks cluster status (resources and service groups), and distributes this information among cluster nodes using the low latency transport layer.



- Low latency transport (LLT): LLT operates in kernel space, supporting communication between servers in a cluster, and handles heartbeat communication. LLT runs directly on top of the DLPI layer in UNIX. LLT load balances cluster communication over the private network links.

A critical question related to cluster communication is, “What happens when communication is lost between cluster servers?” VCS uses heartbeats to determine the health of its peers and requires a minimum of two heartbeat paths, either private, public, or disk based. With only a single heartbeat path, VCS is unable to determine the difference between a network failure and a system failure. The process of handling loss of communication on a single network as opposed to a multiple network is called *jeopardy*. So, if there is a failure on all communication channels, the action taken depends on what channels have been lost and the state of the channels prior to the failure. Essentially, VCS will take action such that only one node has a service group at any one time; in some instances, disabling failover to avoid possible corruption of data. A full discussion is included in “Network partitions and split-brain” in Chapter 13, “Troubleshooting and Recovery”, in the *VERITAS Cluster Server User Guide*.

## Cluster installation and setup

Installation of VCS on AIX is via **installp** or SMIT. It should be noted, however, that if **installp** is used, then LLT, GAB, and the `main.cf` file must be configured manually. Alternatively, the `installvcs` script can be used to handle the installation of the required software and initial cluster configuration.

After the VCS software has been installed, configuration is typically done via the VCS GUI interface. The first step is to carry out careful planning of the desired high availability environment. There are no specific tools in VCS to help with this process. Once this has been done, service groups are created and resources are added to them, including resource dependencies. Resources are chosen from the bundled agents and enterprise agents or, if there are no existing agents for a particular resource, a custom agent can be built. After the service groups have been defined, the cluster definition is automatically synchronized to all cluster servers.

Under VCS, the cluster configuration is stored in ASCII files. The two main files are the *main.cf* and *types.cf*:

- `main.cf`: Defines the entire cluster
- `types.cf`: Defines the resources

These files are user readable and can be edited in a text editor. A new cluster can be created based on these files as templates.

## Cluster administration facilities

Administration in a VERITAS cluster is generally carried out via the cluster manager GUI interface. The cluster manager provides a graphical view of cluster status for resources, service groups, and heartbeat communication among others.

- ▶ Administration security: A VCS administrator can have one of five user categories. These include *Cluster Administrator*, *Cluster Operator*, *Group Administrator*, *Group Operator*, and *Cluster Guest*. Functions within these categories overlaps. The Cluster Administrator has full privileges and the ClusterGuest has read only function. User categories are set implicitly for the cluster by default, but can be also be set explicitly for individual service groups.
- ▶ Logging: VCS generates both error messages and log entries for activity in the cluster from both the cluster engine and each of the agents. Log files related to the cluster engine can be found in the `/var/VRTSsvc/log` directory, and agent log files in the `$VCS_HOME/log` directory. Each VCS message has a tag, which is used to indicate the type of the message. Tags are of the form TAG\_A-E, where TAG\_A is an error message and TAG\_D indicates that an action has occurred in the VCS cluster. Log files are ASCII text and user readable. However, the cluster management interface is typically used to view logs.
- ▶ Monitoring and diagnostic tools: VCS can monitor both system events and applications. Event triggers allow the system administrator to define actions to be performed when a service group or resource hits a particular trigger. Triggers can also be used to carry out an action before the service group comes online or goes offline. The action is typically a script, which can be edited by the user. The event triggers themselves are pre-defined. Some can be enabled by administrators, where others are enabled by default. In addition, VCS provides simple network management protocol (SNMP), management interface base (MIB), and simple mail transfer protocol (SMTP) notification. The severity level of a notification is configurable. Event notification is implemented in VCS using *triggers*.
- ▶ Emulation tools: There are no emulation tools in the current release of VERITAS Cluster Server for AIX Version 2.0.

## HACMP and VERITAS Cluster Server compared

The following section describes HACMP and highlights where terminology and operation differ between HACMP and VERITAS Cluster Server (VCS). HACMP and VCS have fairly comparable function, but differ in some areas. VCS has support for cross-platform management, is integrated with other VERITAS

products, and uses a GUI interface as its primary management interface. HACMP is optimized for AIX and pSeries servers, and is tightly integrated with the AIX operating system. HACMP can readily utilize availability functions in the operating system to extend its capabilities to monitoring and managing of non-cluster events.

## Components of an HACMP cluster

An HACMP cluster is similarly comprised nodes, external shared disk, networks, applications, and clients.

- ▶ **Nodes:** Nodes in an HACMP cluster are called cluster nodes, compared with VCS cluster server. There can be up to 32 nodes in an HACMP/ES cluster, including in a concurrent access configuration. A node will run an application or multiple applications, and can be added to or removed from a cluster dynamically.
- ▶ **Shared external disk devices:** HACMP has built-in support for a wide variety of disk attachments, including Fibre Channel and several varieties of SCSI. HACMP provides an interface for OEM disk vendors to provide additional attachments for NAS, SAN, and other disks.
- ▶ **Networks:** IP networks in an HACMP cluster are used for both heartbeat/message communication, to determine the status of the resources in the cluster, and also for client traffic. HACMP uses an optimized heartbeat protocol over IP. Supported IP networks include Ethernet, FDDI, token-ring, SP-Switch, and ATM. Non-IP networks are also supported to prevent the TCP/IP network from becoming a single point of failure in a cluster. Supported non-IP networks include serial (rs232), target mode SSA (TMSSA), and target mode SCSI (TMSCSI) via the shared disk cabling. Public networks in HACMP carry both heartbeat/message and client traffic. Networks based on X.25 and SNA are also supported as cluster resources. Cluster configuration information is propagated over the public TCP/IP networks in an HACMP cluster. However, heartbeats and messages, including cluster status information, is communicated over all HACMP networks.

## Cluster resources

Resources to be made highly available include network adapters, shared storage, IP addresses, applications, and processes. Resources have a type, and you can have multiple instances of a resource type.

- ▶ **HACMP event scripts:** Both HACMP and VCS support built-in processing of common cluster events. HACMP provides a set of predefined event scripts that handle bringing resources online, taking them offline, and moving them if required. VCS uses bundled agents. HACMP provides an event customization process and VCS provides a means to develop agents.

- Application server: This is the HACMP term used to describe how applications are controlled in an HACMP environment. Each application server is comprised of a start and stop script, which can be customized on a per node basis. Sample start and stop scripts are available for download for common applications at no cost.
- Application monitor: Both HACMP and VCS have support for application monitoring, providing for retry/restart recovery, relocation of the application, and for different processing requirements, based on the node where the application is being run.

The function of an application server coupled with an application monitor is similar to a VCS enterprise agent.

- ▶ Resource group: This is equivalent to a VCS service group, and is the term used to define a set of resources that comprise a service. The type of a resource group defines takeover relationships, which includes:
  - Cascading: A list of participating nodes is defined for a resource group, with the order of nodes indicating the node priority for the resource group. Resources are owned by the highest priority node available. If there is a failure, then the next active node with the highest priority will take over. Upon reintegration of a previously failed node, the resource group will move back to the preferred highest priority node.
    - Cascading without fall back (CWOFF): This is a feature of cascading resource groups which allows a previously stopped cluster node to be reintegrated into a running HACMP cluster without initiating a take back of resources. The environment once more becomes fully highly available and the system administrator can choose when to move the resource group(s) back to the server where they usually run.
    - Dynamic node priority (DNP) policy: It is also possible to set a dynamic node priority (DNP) policy, which can be used at failover time to determine the best takeover node. Each potential takeover node is queried regarding the DNP policy, which might be something like least loaded. DNP uses the Event Management component of RSCT and is therefore available with HACMP/ES only. Obviously, it only makes sense to have a DNP policy where there are more than two nodes in a cluster. Similarly, the use of Load to determine the takeover node in a VCS cluster is only relevant where there are more than two cluster servers. There is an extensive range of possible values that can be used to define a DNP policy; run the **haemqvar -h cluster\_name** command to get a full list.
  - Rotating: A list of participating nodes is defined for a resource group, with the order indicating the node priority for a resource group. When a cluster node is started, it will try to bring online the resource group for which it has the highest priority. Once all rotating resource groups have been brought

online, any additional cluster nodes that participate in the resource group join as standby. Should there be a failure, a resource group will move to an available standby (with the highest priority) and remain there. At reintegration of a previously failed node, there is no take back, and the server simply joins as standby.

- Concurrent: Active on multiple nodes at the same time. Applications in a concurrent resource group are active on all cluster nodes, and access the same shared data. Concurrent resource groups are typically used for applications that handle access to the data, although the cluster lock daemon *cllockd* is also provided with HACMP to support locking in this environment. Raw logical volumes must be used with concurrent resources groups. An example of an application that uses concurrent resource groups is Oracle 9i Real Application Cluster.

In HACMP Version 4.5 or later, resource groups are brought online in parallel by default to minimize the total time required to bring resources online. It is possible, however, to define a temporal order if resource groups need to be brought online sequentially. Other resource group dependencies can be scripted and executed via pre- and post-events to the main cluster events.

HACMP does not have an equivalent to VCS system zones.

## Cluster configurations

HACMP and VCS are reasonably comparable in terms of supported cluster configurations, although the terminology differs. HACMP cluster configurations include:

- ▶ Standby configurations: Support a traditional hardware configuration where there is redundant equipment available as a hot standby. Can have both cascading and rotating resources in a hot standby configuration.
- ▶ Takeover configurations: All cluster nodes do useful work and act as a backup to each other. Takeover configurations include cascading mutual takeover, concurrent, one-to-one, one-to-any, any-to-one, and any-to-any.
- ▶ Concurrent: All cluster nodes are active and have simultaneous access to the same shared resources.

## Cluster communications

Cross cluster communication is a part of all high availability software, and in HACMP this task is carried out by the following components:

- ▶ Cluster manager daemon (*clstrmgr*): This can be considered similar to the VCS cluster engine and must be running on all active nodes in an HACMP

cluster. In the classic feature of HACMP, the clstrmgr is responsible for monitoring nodes and networks for possible failure, and keeping track of the cluster peers. In the enhanced scalability feature of HACMP (HACMP/ES), some of the clstrmgr function is carried out by other components, specifically, the group services and topology services components of RSCT. The clstrmgr executes scripts in response to changes in the cluster (events) to maintain availability in the clustered environment.

- ▶ Cluster SMUX peer daemon (clsmuxpd): This provides cluster based simple network management protocol (SNMP) support to client applications and is integrated with Tivoli Netview via HATivoli in a bundled HACMP plug-in. VCS has support for SNMP. There are two additional HACMP daemons: the cluster lock daemon (cllockd) and cluster information daemon (clinfo). Only clstrmgr and clsmuxpd need to be running in the cluster.
- ▶ Reliable scalable cluster technology (RSCT): This is used extensively in HACMP/ES for heartbeat and messaging, monitoring cluster status, and event monitoring. RSCT is part of the AIX 5L base operating system and is comprised of:
  - Group services: Co-ordinates distributed messaging and synchronization tasks.
  - Topology services: Provides heartbeat function, enables reliable messaging, and co-ordinates membership of nodes and adapters in the cluster.
  - Event management: Monitors system resources and generates events when resource status changes.

HACMP and VCS both have a defined method to determine whether a remote system is alive, and a defined response to the situation where communication has been lost between all cluster nodes. These methods essentially achieve the same result, which is to avoid multiple nodes trying to grab the same resources.

## Cluster installation and setup

Installation of HACMP for AIX software is via the standard AIX install process using **installp**, from the command line or via SMIT. Installation of HACMP will automatically update a number of AIX files, such as /etc/services and /etc/inittab. No further system related configuration is required following the installation of the HACMP software.

The main smit HACMP configuration menu (fast path **smitty hacmp**) outlines the steps required to configure a cluster. The cluster topology is defined first and synchronized via the network to all nodes in the cluster and then the resource groups are set up. Resource groups can be created on a single HACMP node and the definitions propagated to all other nodes in the cluster. The resources,

which comprise the resource group, have implicit dependencies that are captured in the HACMP software logic.

HACMP configuration information is held in the object data manager (ODM) database, providing a secure but easily shareable means of managing the configuration. A *cluster snapshot* function is also available, which captures the current cluster configuration in two ASCII user readable files. The output from the snapshot can then be used to clone an existing HACMP cluster or to re-apply an earlier configuration. In addition, the snapshot can be easily modified to capture additional user-defined configuration information as part of the HACMP snapshot. VCS does not have a snapshot function *per se*, but allows for the current configuration to be dumped to file. The resulting VCS configuration files can be used to clone cluster configurations. There is no VCS equivalent to applying a cluster snapshot.

## Cluster administration facilities

Cluster management is typically via the System Management Interface Tool (SMIT). The HACMP menus are tightly integrated with SMIT and are easy to use. There is also close integration with the AIX operating system.

- ▶ **Administration security:** HACMP employs AIX user management to control access to cluster management function. By default, the user must have root privilege to make any changes. AIX *roles* can be defined if desired to provide a more granular level of user control. Achieving high availability requires good change management, and this includes restricting access to users who can modify the configuration.
- ▶ **Logging:** HACMP log files are simple ASCII text files. There are separate logs for messages from the cluster daemons and for cluster events. The primary log file for cluster events is the `hacmp.out` file, which is by default in `/tmp`. The system administrator can define a non default directory for individual HACMP log files. The contents of the log files can be viewed via SMIT or a Web browser. In addition, RSCT logs are also maintained for HACMP/ES.
- ▶ **Monitoring and diagnostic tools:** HACMP has extensive event monitoring capability based on the RSCT technology, and it is possible to define a custom HACMP event to run in response to the outcome of event monitoring. In addition, multiple pre- and post-events can be scripted for all cluster events to tailor them for local conditions. HACMP and VCS both support flexible notification methods, SNMP, SMTP, and e-mail notification. HACMP uses the AIX error notification facility and can be configured to react to any error reported to AIX. VCS is based on event triggers and reacts to information from agents. HACMP also supports pager notification.
- ▶ **Emulation tools:** Actions in an HACMP cluster can be emulated. There is no emulation function in VCS.

Both HACMP and VCS provide tools to enable maintenance and change in a cluster without downtime. HACMP has the cluster single point of control (CSPOC) and dynamic reconfiguration capability (DARE). CSPOC allows a cluster change to be made on a single node in the cluster and for the change to be applied to all nodes. Dynamic reconfiguration uses the `clldare` command to change configuration, status, and location of resource groups dynamically. It is possible to add nodes, remove nodes, and support rolling operating system or other software upgrades. VCS has the same capabilities and cluster changes are automatically propagated to other cluster servers. However, HACMP has the unique ability to emulate migrations for testing purposes.

## HACMP and VERITAS Cluster Server feature comparison summary

Table D-1 provides a high level feature comparison of HACMP and VERITAS Cluster Server, followed by Table D-2 on page 338, which compares support hardware and software environments. It should be understood that both HACMP and VERITAS Cluster Server have extensive functions that can be used to build highly available environments and the online documentation for each product must be consulted.

*Table D-1 HACMP/VERITAS Cluster Server feature comparison*

| Feature                            | HACMP  | VCS for AIX  |
|------------------------------------|--|--|
| Resource/service group failover.   | Yes, only affected resource group moved in response to a failure. Resource group moved as an entity. | Yes, only affected service group moved in response to a failure. Service group moved as an entity. |
| IP address takeover.               | Yes.   | Yes.   |
| Local swap of IP address.          | Yes.   | Yes.   |
| Management interfaces.             | CLI and SMIT menus.  | CLI, Java-based GUI, and Web console.  |
| Cross-platform cluster management. | No.  | Yes, but with the requirement that nodes in a cluster be homogenous.                               |
| Predefined resource agents.        | N/A. Management of resources integrated in the logic of HACMP.                                       | Yes.   |
| Predefined application agents.     | No. Sample application server start/stop scripts available for download.                             | Yes, Oracle, DVB2, and VVR.  |



| Feature   | HACMP   | VCS for AIX  |
|---|---|--|
| Automatic cluster synchronization of volume group changes.            | Yes.  | N/A.   |
| Ability to define resource relationships.                             | Yes, majority of resource relationships integral in HACMP logic. Others can be scripted.          | Yes, via CLI and GUI.  |
| Ability to define resource/service group relationships.               | Yes, to some extent via scripting.  | Yes, via CLI and GUI.  |
| Ability to decide fail-over node at time of failure based on load.    | Yes, dynamic node priority with cascading resource group. Number of ways to define load via RSCT. | Yes, load option of fallover service group. Single definition of load. |
| Add/remove nodes without bringing the cluster down.                   | Yes.  | Yes.   |
| Ability to start/shutdown cluster without bringing applications down. | Yes.  | Yes.   |
| Ability to stop individual components of the resource/service group.  | No.   | Yes.   |
| User level security for administration.                               | Based on the operating system with support for roles.   | Five security levels of user management.                               |
| Integration with backup/recovery software.                            | Yes, with Tivoli Storage Manager.   | Yes, with VERITAS NetBackup.   |
| Integration with disaster recovery software.                          | Yes, with HAGEO.  | Yes, with VERITAS Volume Replicator and VERITAS Global Cluster Server. |
| Emulation of cluster events.  | Yes.  | No.  |

Table D-2 HACMP/VERITAS Cluster Server environment support

| Environment                              | HACMP   | VCS for AIX   |
|--|---|---|
| Operating system                         | AIX 4.X/5L 5.1.   | AIX 4.3.3/5L 5.1. VCS on AIX 4.3.3 uses AIX LVM, JFS/JFS2 only.   |
| Network connectivity                     | Ethernet (10/100 Mbs), Gigabit Ethernet, ATM, FDDI, Token-ring, and SP switch.  | Ethernet (10/100 Mbs) and Gigabit Ethernet.   |
| Disk connectivity                        | SCSI, Fibre Channel, and SSA.   | SCSI, Fibre Channel, and SSA.   |
| Maximum servers in a cluster             | 32 with HACMP Enhanced Scalability (ES) feature, eight with HACMP feature.  | 16.   |
| Maximum servers - Concurrent disk access | 32 - Raw logical volumes only.  | N/A.  |
| LPAR support                             | Yes.  | Yes.  |
| SNA                                      | Yes.  | No.   |
| Storage subsystems                       | See <i>HACMP Version 4.5 for AIX Release Notes</i> available for download at <a href="http://www.ibm.com/woi">http://www.ibm.com/woi</a> . Search for 5765-E54. | See <i>VERITAS Cluster Server 2.0 for AIX Release Notes</i> , available for download from <a href="http://support.veritas.com">http://support.veritas.com</a> . |

# Abbreviations and acronyms

|               |  |             |  |
|---------------|--|-------------|--|
| <b>AIX</b>    | Advanced Interactive Executive                               | <b>IBM</b>  | International Business Machines Corporation  |
| <b>APAR</b>   | Authorized Program Analysis Report                           | <b>ILS</b>  | IBM Learning Services                        |
| <b>API</b>    | Application Program Interface                                | <b>IP</b>   | Internet Protocol                            |
| <b>ASCII</b>  | American National Standards Code for Information Interchange | <b>ITSO</b> | International Technical Support Organization |
| <b>ASL</b>    | Array Support Library  | <b>JFS</b>  | Journaled File System                        |
| <b>ATM</b>    | Asynchronous Transfer Mode                                   | <b>KDB</b>  | Kernel Debugger                              |
| <b>BMR</b>    | Bare Metal Restore   | <b>LDAP</b> | Lightweight Directory Access Protocol        |
| <b>CD-ROM</b> | Compact Disc Read Only Memory                                | <b>LLT</b>  | Low Latency Transport                        |
| <b>CLI</b>    | Command Line Interface                                       | <b>LP</b>   | Logical Partition                            |
| <b>CSPOC</b>  | Cluster Single Point of Control                              | <b>LPP</b>  | Licensed Product Package                     |
| <b>CWOF</b>   | Cascading Without Fallback                                   | <b>LUN</b>  | Logical Unit Number                          |
| <b>DARE</b>   | Dynamic Reconfiguration                                      | <b>LV</b>   | Logical Volume                               |
| <b>DCO</b>    | Data Change Object   | <b>LVCB</b> | Logical Volume Control Block                 |
| <b>DDL</b>    | Device Discovery Layer                                       | <b>LVM</b>  | Logical Volume Manager                       |
| <b>DGID</b>   | Disk Group ID  | <b>MIB</b>  | Management Interface Base                    |
| <b>DMP</b>    | Dynamic MultiPathing   | <b>MWC</b>  | Mirror Write Cache                           |
| <b>DNP</b>    | Dynamic Node Priority  | <b>MWCC</b> | Mirror Write Consistency Checking            |
| <b>DRL</b>    | Dirty Region Logging   | <b>NAS</b>  | Network-Attached Storage                     |
| <b>ESS</b>    | Enterprise Storage Server                                    | <b>NFS</b>  | Network File System                          |
| <b>FDDI</b>   | Fiber Distributed Data Interface                             | <b>NIM</b>  | Network Installation Management              |
| <b>FMR</b>    | Fast Mirror Resynchronization                                | <b>ODM</b>  | Object Data Manager                          |
| <b>FTP</b>    | File Transfer Protocol                                       | <b>PCI</b>  | Peripheral Component Interface               |
| <b>GAB</b>    | Group Membership Services/Atomic Broadcast                   | <b>PP</b>   | Physical Partition                           |
| <b>GUI</b>    | Graphical User Interface                                     | <b>PV</b>   | Physical Volume                              |
| <b>HACMP</b>  | High Availability Cluster Multi-Process                      | <b>PVID</b> | Physical Volume Identifier                   |
| <b>HAD</b>    | High Availability Daemon                                     | <b>RAID</b> | Redundant Array of Independent Disks         |
|               |  | <b>RAM</b>  | Random Access Memory                         |

|             |                                       |
|-------------|---------------------------------------|
| <b>ROM</b>  | Read Only Memory                      |
| <b>RSCT</b> | RISC System Cluster<br>Technology     |
| <b>SAN</b>  | Storage Area Network                  |
| <b>SCSI</b> | Small Computer System<br>Interface    |
| <b>SDD</b>  | Subsystem Device Driver               |
| <b>SMIT</b> | System Management<br>Interface Tool   |
| <b>SMTP</b> | Simple Mail Transfer Protocol         |
| <b>SNA</b>  | System Network Architecture           |
| <b>SNMP</b> | Simple Network Management<br>Protocol |
| <b>SSA</b>  | Serial Storage Architecture           |
| <b>TCP</b>  | Transmission Control Protocol         |
| <b>UDP</b>  | User Datagram Protocol                |
| <b>UFS</b>  | UNIX File System                      |
| <b>VCS</b>  | VERITAS Cluster Server                |
| <b>VEA</b>  | VERITAS Enterprise<br>Administrator   |
| <b>VGDA</b> | Volume Group Descriptor<br>Area       |
| <b>VGID</b> | Volume Group Identifier               |
| <b>VGRA</b> | Volume Group Reserved Area            |
| <b>VGSA</b> | Volume Group Status Area              |
| <b>VMM</b>  | Virtual Memory Management             |
| <b>VVR</b>  | VERITAS Volume Replicator             |
| <b>VxFS</b> | VERITAS File System                   |
| <b>VxVM</b> | VERITAS Volume Manager                |
| <b>XOR</b>  | Exclusive OR                          |

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 344.

- ▶ *AIX 5L Differences Guide Version 5.1 Edition*, SG24-5765
- ▶ *AIX Logical Volume Manager from A to Z: Introduction and Concepts*, SG24-5432
- ▶ *AIX Logical Volume Manager from A to Z: Troubleshooting and Commands*, SG24-5433
- ▶ *AIX Reference for Sun Solaris Administrators*, SG24-6584
- ▶ *Problem Solving and Troubleshooting in AIX 5L*, SG24-5496

## Other resources

These publications are also relevant as further information sources:

- ▶ Massiglia, *VERITAS in E-Business*, VERITAS Software Corporation (internal VERITAS publication only)
- ▶ *PCI Adapter Placement Guide*, found at:  
[http://www.ibm.com/servers/eserver/pseries/library/hardware\\_docs/](http://www.ibm.com/servers/eserver/pseries/library/hardware_docs/)
- ▶ *Quick Reference: AIX Logical Volume Manager and VERITAS Volume Manager*, found at:  
[http://www.ibm.com/servers/aix/products/aixos/whitepapers/lvm\\_ver.html](http://www.ibm.com/servers/aix/products/aixos/whitepapers/lvm_ver.html)
- ▶ *Quick Reference: AIX Journaled File Systems and VERITAS File System*, found at:  
[http://www.ibm.com/servers/aix/products/aixos/whitepapers/jfs\\_vfs.html](http://www.ibm.com/servers/aix/products/aixos/whitepapers/jfs_vfs.html)
- ▶ *VERITAS Bundled Agents Reference Guide*
- ▶ *VERITAS Cluster Server Agents Developers Guide*
- ▶ *VERITAS Cluster Server User Guide*

- ▶ *VERITAS File System 3.4 - Administrator's Guide (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246771.htm>
- ▶ *VERITAS File System 3.4 - Installation Guide (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246772.htm>
- ▶ *VERITAS File System 3.4 - Release Notes (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246773.htm>
- ▶ *VERITAS Foundation Suite and Foundation Suite HA 3.4 - Release Notes (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246770.htm>
- ▶ *VERITAS Volume Manager 3.2 - Administrator's Guide (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246774.htm>
- ▶ *VERITAS Volume Manager 3.2 - Hardware Notes (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246775.htm>
- ▶ *VERITAS Volume Manager 3.2 - Installation Guide (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246776.htm>
- ▶ *VERITAS Volume Manager 3.2 - Migration Guide (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246777.htm>
- ▶ *VERITAS Volume Manager 3.2 - Release Notes (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246778.htm>
- ▶ *VERITAS Volume Manager 3.2 - Troubleshooting Guide (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246779.htm>
- ▶ *VERITAS Volume Manager 3.2 VERITAS Enterprise Administrator User's Guide (AIX)*, found at:  
<http://seer.support.veritas.com/docs/246780.htm>

All VERITAS documentation can be found at:

<http://seer.support.veritas.com>

## Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ AIX 5L Documentation  
[http://publibn.boulder.ibm.com/cgi-bin/ds\\_form?lang=en\\_US&viewset=AIX/](http://publibn.boulder.ibm.com/cgi-bin/ds_form?lang=en_US&viewset=AIX/)

- ▶ AIX Toolbox for Linux Applications  
<http://www.ibm.com/servers/aix/products/aixos/linux/index.html>
- ▶ IBM Web site  
<http://www.ibm.com/>
- ▶ IBM Learning Services  
<http://www.ibm.com/services/learning/>
- ▶ IBM Offering Information  
<http://www.ibm.com/woi/>
- ▶ IBM Solution Partnership Centers  
<http://www.developer.ibm.com/spc/index.html>
- ▶ Required AIX filesets that must be installed prior to installing VERITAS Foundation Suite and Foundation Suite HA 3.4  
<http://seer.support.veritas.com/docs/245873.htm>
- ▶ VERITAS e-mail Notification Subscription Utility  
<http://maillist.support.veritas.com/subscribe.asp>
- ▶ VERITAS Enabled - Compatibility Lists  
[http://www.veritas.com/enabled/compatibility\\_lists.html](http://www.veritas.com/enabled/compatibility_lists.html)
- ▶ VERITAS ftp site to download VxExplore  
<ftp://ftp.veritas.com/pub/support/vxexplore.tar.Z>
- ▶ VERITAS Peer to Peer News Groups  
<http://news.support.veritas.com/>
- ▶ VERITAS Software  
<http://www.veritas.com/>
- ▶ VERITAS Software Alert  
[http://support.veritas.com/prodlist\\_path\\_alerts.htm](http://support.veritas.com/prodlist_path_alerts.htm)
- ▶ VERITAS Software Beta Program  
<http://beta.veritas.com/>
- ▶ VERITAS Support  
<http://support.veritas.com/>
- ▶ vLicense - Managing Your License Keys  
<http://vlicense.veritas.com/>
- ▶ pSeries Hardware Documentation  
[http://www.ibm.com/servers/eserver/pseries/library/hardware\\_docs/](http://www.ibm.com/servers/eserver/pseries/library/hardware_docs/)

- ▶ pSeries Resource Library  
<http://www.ibm.com/servers/eserver/pseries/library/>
- ▶ pSeries Technical Support  
<http://techsupport.services.ibm.com/server/support?view=pSeries/>
- ▶ pSeries Technical Support - Fix Delivery Center for AIX 5L Version 5.1  
<http://techsupport.services.ibm.com/server/aix.fdc51/>
- ▶ pSeries Technical Support - Hardware Technical Mailings  
<http://techsupport.services.ibm.com/rs6k/mcode.html>
- ▶ pSeries Technical Support - Software Technical Mailings  
<http://techsupport.services.ibm.com/server/listserv/>
- ▶ pSeries Technical Support - Tips and howto  
<http://techsupport.services.ibm.com/server/nav?fetch=tah/>
- ▶ pSeries Technical Support - Tips for AIX Administrators  
<http://techsupport.services.ibm.com/server/aix.techTips/>
- ▶ pSeries Technical Support - Troubleshooting  
<http://techsupport.services.ibm.com/server/nav?fetch=ts/>

## How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

## IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.



# Index

## Symbols

.cshrc 70  
.kshrc 70  
/etc/default/vxassist 98  
/etc/dumpdates 163  
/etc/filesystems 120, 123, 151  
/etc/init.d 70, 79  
/etc/inittab 70, 79, 186  
/etc/objrepos 182  
/etc/profile 70  
/etc/rc.d/rc2.d 70  
/etc/vx/bin 70, 83  
/etc/vx/cntrl.exclude file 57  
/etc/vx/diag.d/vxdmpinq 65  
/etc/vx/disks.exclude 57  
/etc/vx/dmp 132  
/etc/vx/enclr.exclude 57  
/etc/vx/sparelist 168  
/etc/vx/tempdb 196–197  
/opt/VRTS/bin 70  
/sbin 70  
/tmp/VRTSinstall.out 294  
/usr/lib/drivers 187  
/usr/sbin 70  
/var/vxvm/ 196

## Numerics

32-bit kernel 210  
64-bit kernel 190, 210  
64-bit mode enabled 291

## A

Active/active disk arrays 17  
Active/passive disk arrays 17  
Agents 325  
AIX  
    bonus pack CD 52, 71, 84  
    Field Technical Specialists 5  
    kernel trace command 190  
    Logical Volume Manager 89  
    tech tips 289  
alliance teams 5

allocation policy 201  
    notstrict 201  
    strict 201  
    super strict 201  
Application monitor 332  
Application server 332  
Array Support Libraries (ASLs) 192  
Asynchronous I/O 217  
Attributes 327

## B

Backup 206–207  
backup command 164  
Beta Programs 286  
binary-tree 210  
Block 195  
block size 120  
block-allocation method 36  
Bonus Pack 4, 6  
bootinfo -K 190  
B-tree 8  
Bundled agents 325

## C

C shell 70  
cache advisories 38, 216  
Caching 216  
centralized management 13, 35  
cfgmgr 182, 199  
change disk groups to AIX volume groups 295  
changing e-mail destination 168  
Changing volume layouts 149  
check\_all 135  
check\_disabled 135  
chfs 215  
chpv 46, 89  
clr\_disk\_vm\_owner() 185  
cluster 324  
cluster configurations 328  
cluster node 331  
Cluster resources 325  
cluster servers 324  
Coexistence of JFS/JFS2 and VxFS 218

- Command line interface (CLI) 83, 87, 125
- comparison with other platforms 182
- concatenated volume 98–99, 101
- concatenation 25
- Configuration copy 196
- consolidation 72
- controller 66
- cpio 72
- crash 191
- creating a new VERITAS File System 296
  - /etc/filesystems 296
  - crfs 296
  - mkfs 296
- creating a volume layout 297
  - /etc/default/vxassist 297
- creating disk groups 295
- Creating volumes 98
- crfs 120, 164
- Custom agents 325
- Custom Installation 56–58, 60

## D

- daemons 70
- DASD sector 195
- data availability 2
- Data Change Object (DCO) logs 143
- dd 72
- decrease file system size 215
- default block size 120
- default inode size 37
- defragfs 216
- defragmentation 2, 39
- delayed writes 210
- demo license 7
- deporting 153
- Device discovery 17, 199
- device discovery layer (DDL) 192, 199
- device path for a VM disk 194
- df 151
- df -v 213
- direct extents 37
- Direct I/O 38, 210, 216
  - performance 217
- dirty bit 34
- Dirty Region Logging (DRL) 33, 143, 204
- Discovered direct I/O 38, 217
- discovered\_direct\_iosz parameter 38
- disk array 14

- Disk channel 324
- disk enclosures 46
- disk group 20, 87, 112, 115, 195
- disk group ID (DGID) 195
- disk layout 35
- disk media name 93
- Disk states
  - error 113
  - LVM 113
  - offline 113
  - online 113
  - online invalid 113
- disks 87
- dump level 163
- Dynamic disk group 87
- Dynamic Multipathing (DMP) 16, 42, 66, 132, 192
  - supported arrays 132

## E

- encapsulation 46
- enclosure-based naming 18, 184
- enclosures 56, 87
- Enhanced Journaled File System (JFS2) 8, 209
- Enterprise agents 325
- Enterprise Storage Server (ESS) 181, 207
- errno 28 295
- error daemon 135
- evacuating volumes from a disk 169
- Evacuation 72
- evaluation version 4, 6
- event trigger 330
- excluding disks from hot relocation 167
- exclusive OR (XOR) 30
- extend a VERITAS file system 295
  - completely full 295
- extended attribute record 215
- extent 36
  - address 36
  - length 36
- Extent-based allocation 36

## F

- failed disks
  - replacing and removing 171
- Fast Mirror Resynchronization (FMR) 291
- FastResync 204, 291
- file systems 22, 87, 120
  - block sizes 36

- defragmenting 178
- increasing 125
- metadata 212
- reducing 125
- resizing 39
- setting block size 175
- setting intent log size 175
- snapshots 156
- FlashCopy 207
- fsadm 125–126, 215
  - defragmenting file systems 178
- fsck 38, 212
- fsync 210
- functional comparison 181

## G

- GABdisk 325
- Graphical User Interface (GUI) 84
- Group Membership Services/Atomic Broadcast (GAB) 324, 328
- growing a file system 215

## H

- HACMP 324, 330
- HACMP event scripts 331
- hardware failures 297
- heartbeat protocol 331
- heterogeneous environment 2
- high availability 323
- High Availability Daemon (HAD) 328
- High Availability features 12
- High Performance and Scalability features 12, 34
- host access 197
- hot relocation 33, 80, 129, 166
- Hot spare management 208
- hot sparing 80, 129
  - described 166
  - enabling in favour of hot relocation 168

## I

- I/O balancing 17
- IBM 7133 SSA array 192
- IBM Enterprise Storage Server (ESS) 193
- IBM Global Services 4
- importing 153
- importing a disk group 197
- indirect extents 37

- initial inode list 37
- inittab 70
- inode list 37
- Inodes 37, 212
  - data structure 37
  - default inode size 37
- Installation
  - APARS 43
  - hardware 42
  - licensing 42
  - software 42
- Installation issues 290, 292
  - VRTSinstall issues 292
  - VRTSob and VRTSobgui packages 293
  - vxinstall 293
- installp 47, 71, 73
- installvcs script 329
- Integration 13, 35
- intent log 38, 212
  - circular 212
  - default size 212
- internal volume 33
- interoperability 181
- ioctl 38, 216

## J

- jeopardy 329
- JFS 7, 209
  - compressed file systems 216
  - inode structure 213
  - maximum file size 210
- JFS commands 305
- JFS2 209
  - commands 305
  - inode structure 213
  - maximum file size 210
- journal log 212
  - circular 212
  - default size 212
- journaling 2, 38, 212
- jre.tar 293

## K

- K50isid 79
- kernel buffer 216
- Kernel Debugger (KDB) 191

## L

- layered volume 33, 147
- layout 101
- LAYOUT column 118
- LENGTH column 118
- license key 46, 53–54
- licensed product packages (LPPs) 189
- load balancing 16
- Logging 32
- logging metadata 212
- logical partition (LP) 198
- Logical Unit Number (LUN) 193
- logical volume 120, 198
- Logical volume concepts 194
- Logical Volume Manager 7, 299
- Low Latency Transport (LLT) 324, 329
- lppchk 189
- lsdev 64
- lspp 189
- lspv 46, 75, 89
- LVM 46
- LVM and VxVM logical and physical components 199
- LVM commands 300

## M

- main.cf 329
- Man pages 70
- Management Console 85
- MANPATH 70, 294
- marking a disk as a hot spare 167
- metadata 38
  - logging 38
- metanode 132
- Methods of administration 77
- mirror copy failure 202
- mirror write cache (MWC) 203
- Mirror write consistency checking (MWCC) 203
- Mirrored (RAID-1) volumes 98
- mirrored and striped volumes 110
- Mirrored and striped volumes (RAID 1+0) volumes 98
- mirrored copies 200
- mirrored volumes 104
- Mirroring 26, 200
  - data redundancy 27
  - read/write policy 201
- Mirroring with striping (RAID 1+0) 29

- mirrorvrg 205
- mkfs 120, 187
- mksysb 207
- Monitoring tasks 136
- mount 123, 164
- Mount options
  - blkclear 123
  - convosync 123
  - delaylog 122
  - largefiles 123
  - log 122
  - mincache 123
  - nodatainlog 122
  - nolargefiles 123
  - qllog 123
  - specifying 123
  - tmplog 122
- multipath disk array 16
- multipathing 58, 60, 63, 66
- multipathing policies
  - Active/Active 132
  - Active/Passive 132

## N

- Network channels 324
- Network partitions 329
- newsgroups, IBM 289
- newsgroups, VERITAS 285
- Nodes 324
- nopriv 184
- number of columns 206
- number of inodes 213

## O

- Object Data Manager (ODM) 182, 192, 196
- ODMDIR 182
- odmget 182
- Online defragmentation 39, 215
- Online file system resizing 39, 215
- Online management 2
- Online relay layout 33, 207
- Online volume management 2
- OTHER\_DISKS 56, 63, 66

## P

- Packages
  - VRTSvlic 47

- VRTSvxfs 47
- parity 30
- parity check 108
- patches, how to get 280
  - IBM 283
  - VERITAS 280
- path failover 17
- pathgroup 66
- PersistentResync 205
- physical disk 14, 19, 194
- physical objects 13
- physical partition (PP) 195
- physical volume (PV) 194
- physical volume identifier 193
- Planning
  - for installation 41
- plex 21–22, 198
  - mirrored 27
  - striping 25
- plex copy 204
- private network 325
- private region 89, 196
- Product ID 65, 68
- profile 70
- ps 69, 134
- public network 325, 331

## Q

- qry\_disk\_vm\_owner() 185
- Quick I/O for databases 180
- Quick Installation 56–58, 60
- Quick log 180
- quotas
  - described 176
  - editing 176
  - group hard limit 177
  - group soft limit 177
  - time limit 177
  - turning off 178
  - turning on 176
  - users hard limit 177
  - users soft limit 177
  - viewing quota information 177

## R

- RAID 2, 24, 199
- RAID 0 2, 25
- RAID 0+1 2, 28

- RAID 1 2, 26, 33, 104
- RAID 1+0 2, 29, 99, 110
- RAID 10 29
- RAID 5 2, 30, 33, 109, 200, 204
  - data recovery 31
- RAID 5 log 108
- RAID 5 logs 143
- RAID 5 volume 98, 108, 290
  - 64-bit kernel enabled 291
- raw logical volume 206
- rc scripts 186
- read-ahead algorithm 217
- Redbooks Web site 344
  - Contact us xx
- removing a VERITAS volume 296
  - /etc/filesystems 296
- removing volumes 151
- rename an enclosure 56
- renaming a disk group 156, 295
  - /etc/filesystems 295
- renaming volumes 150
- reserved area 196
- Resizing of a VxFS file system 185
- Resource categories 326
  - On-Off 326
  - On-Only 326
  - Persistent 326
- Resource group 332
  - Cascading 332
  - Cascading without fall back (CWOFF) 332
  - Concurrent 333
  - Dynamic node priority (DNP) policy 332
  - Rotating 332
- resource group 333
- restore daemon 135
- Resynchronization 202
- rmdev 188
- rmfs 151, 296
- rootability 297
- rootdg 46, 73, 93, 98
- rootvg 46, 207

## S

- S50isisd 70, 79
- S94vxnm-host\_infod 70, 79
- S94vxnm-vxnetd 70, 79
- S95vxvm-recover 70, 79
- S96vradmind 70, 79

- S96vxrsyncd 70, 79
- Scripts
  - /etc/rc.d/rc 79
- SCSI commands 192
- Sector 0 186
- Sector 7 185
- Sector 70 186
- Service group 326
- service group dependencies 327
- service group type 326
  - Failover 326
  - Parallel 327
- set\_disk\_vm\_owner() 185
- Shared external disk devices 324, 331
- shared storage 200
- shutdown 79, 81
- shutdown script 186, 294
- Sliced disks 184
- SMIT
  - adding a mirror to a volume 143
  - adding and deleting dirty region logs 144
  - adding and removing RAID5 logs 145
  - adding data change object logs 146
  - adding disks to disk group 153
  - adding mirrors to volumes 142
  - creating concatenated volumes 101
  - creating file systems 121
  - creating mirrored volumes 105
  - creating RAID-5 volumes 109
  - creating striped and mirrored volumes 107
  - creating striped volumes 103
  - deporting disk groups 156
  - described 83
  - listing disk group information 117
  - listing disk information 115
  - listing volumes information 118
  - mounting a file system 124
  - removing data change object logs 147
  - removing disk groups 153
  - removing disks 175
  - removing volumes 152
  - renaming volumes 151
  - resizing a file system 128
- SMIT object classes 183
- snapshot 206
- split mirrored backups 160
- split-brain 329
- ssar 184
- status area 87

- Storage agents 325
- Storage Checkpoints 180
- stripe size 205
- stripe unit size 25
- striped and mirrored volumes 98, 106–107
- striped pro 87, 112
- striped volumes 98, 102
- Striping 25, 205
- striping with mirroring 28
  - fault tolerance 29
- Striping with parity (RAID 5) 30
- structural files 35
- structural fileset 35
- Sub System Device Driver 193
- sub volumes 147
- subdisk 21
- Subsystem Device Driver (SDD) 193
- support agreement 5
- supported network type 325
- Supported VxVM disk types 184
- suppress paths from VxVM's view 63
- Synchronous I/O 217
- system boot 79
- system dump analysis 191
- System zones 328

## T

- tar 72
- task tag 136
- TechAlerts 286
- TechAlerts, VERITAS 280
- technical support 285
  - IBM 288
  - VERITAS 285
- Tivoli Storage Manager 208
- trace 190
- trace hooks 191
- trcrpt 190
- trcstop 190
- trigger 330
- truncation method 215
- types.cf 329

## U

- ucfgvxvm 188
- unconfigure a VERITAS device driver 188

## V

### VEA 130

- adding a mirror to a volume 142
  - adding and removing dirty region logs 144
  - adding and removing RAID5 logs 145
  - adding disks to disk group 153
  - adding disks using 90
  - adding mirrors to volumes 142
  - changing volume layouts 150
  - comparison 88
  - create concatenated volumes 100
  - creating disk groups 93
  - creating file systems 120
  - creating mirrored and striped volumes 112
  - creating striped and mirrored volumes 107
  - deporting disk groups 155
  - described 87
  - disabling controllers 136
  - evacuating volumes from a disk 171
  - excluding disk from hot relocation 168
  - file system snapshots 157
  - finding disk information 114
  - listing disk group information 116
  - listing volume information 118
  - mounting file systems 123
  - removing volumes 152
  - renaming volumes 151
  - replacing disks 174
  - resizing file systems 127
  - volume snapshots 160
- VEA backend server 70, 80
- VEA java GUI 83
- VEA menu bar
- Actions 87
  - File 87
  - Help 87
  - Tools 87
- Vendor ID 65, 68
- VERITAS Bare Metal Restore (BMR) 207
- VERITAS Cluster Server 324
- VERITAS Cluster Server Agent for DB2 on AIX 3
- VERITAS Cluster Server Agent for Oracle on AIX 3
- VERITAS Cluster Server for AIX (VCS) 3
- VERITAS Database Edition for DB2 on AIX 3
- VERITAS Database Edition for Oracle on AIX 3
- VERITAS Database Edition/HA for DB2 on AIX 3
- VERITAS Database Edition/HA for Oracle on AIX 3
- VERITAS device drivers 182
- VERITAS Education Services 5

- VERITAS Enabled Arrays 192
- VERITAS Enterprise Administrator (VEA) 2, 11, 70
- VERITAS File System 2, 11, 34
- VERITAS File System 3.4 Administrator's Guide (AIX) 305
- VERITAS Foundation Suite for AIX 2
- VERITAS Foundation Suite for Solaris 184
- VERITAS Knowledge Base 285
- VERITAS NetBackup 2, 4, 208
- VERITAS NetBackup Agent for DB2 on AIX 3
- VERITAS NetBackup Agent for Oracle on AIX 3
- VERITAS Services 4
- VERITAS support 71
- VERITAS TechAlerts 280
- VERITAS Volume Manager 2, 11
- VERITAS Volume Manager 3.2 Troubleshooting Guide (AIX) 297
- VERITAS Volume Manager Administrator's Guide 207
- VERITAS Volume Replicator 80
- VERITAS Volume Replicator (VVR) for AIX 3, 80
- VeritasVolumes 89
- Version 4 disk layout 35
- virtual object 19, 22
- VM disk 19–21
- volume 22
- snapshots 156
- Volume Group Descriptor Area (VGDA) 196
- Volume Group Identifier (VGID) 195
- Volume Group Reserved Area (VGRA) 196
- Volume Group Status Area (VGSA) 196, 202
- volume groups 195
- Volume kernel states
- Detached 117
  - Disabled 117
  - Enabled 117
- volume layouts 24, 200
- Mirroring (RAID 1) 26
  - Mirroring with striping (RAID 1+0) 29
  - Striping (RAID 0) 25
  - Striping with mirroring (RAID 0+1) 28
  - Striping with parity (RAID 5) 30
- Volume states
- Active 117
  - Clean 117
  - Empty 117
  - Needsync 117
  - Replay 117
  - Sync 117

- Volume usage type
  - fsgen 138
  - gen 138
  - RAID5 138
- volumes 87
- vrtsadm 85
- VRTSexplorer 71, 286
  - Creating the VRTSexplorer tar file 287
  - Downloading VRTSexplorer from the ftp site 287
  - Installing VRTSexplorer from the CD 287
- VRTSfspro 47–48
- VRTSinstall 49
  - 32 bit 49
  - installation option 47
  - installing using the script 48
- VRTSob 47–48
- VRTSob package installation 290
- VRTSobgui 47–48, 71
- VRTSobgui.msi 84
- VRTSvm doc 47
- VRTSvmman 47
- VRTSvmpro 47–48
- VRTSvxvm 47
- vxassist 206
  - adding a mirror to a volume 140
  - adding a RAID5 log 144
  - adding data change object logs 146
  - adding dirty region logs 143
  - changing the layout of a volume 149
  - comparison with vxmake 98
  - convert option 149
  - creating concatenated volumes 99
  - creating mirrored and striped volumes 112
  - creating mirrored volumes 105
  - creating RAID-5 volumes 109
  - creating snapshots 158
  - creating striped and mirrored volumes 107
  - creating striped volumes 103
  - increasing file systems 125
  - relayout option 149
  - removing dirty region logs 144
  - snapback 159
  - snapclear 160
- vxassist error messages 298
- vxconfigd 69, 80–81, 192, 294
- vxconfigd error messages 298
- vxdc 146
- vxddladm 132, 192
- vxdbg 83, 116
  - adding disk to disk group 152
  - creating disk groups 93
  - deporting disk groups 153
  - importing disk groups 153
  - listing disk group information 115
  - removing disk groups 153
  - removing disks from disk groups 153
  - renaming a disk group 156
- vxdisk 89, 113
  - list disks in a disk group 152
  - list disks in deported disk groups 154
  - listing DMP information 132
  - listing failed disks 171
- vxdiskadd 90, 96
- vxdiskadm 183
  - adding disks 90
  - comparison with vxdiskadd for creating disk groups 96
  - creating disk groups 95
  - described 87
  - disabling DMP 133
  - evacuating volumes from a disk 169, 171
  - excluding a disk from hot relocation 167
  - listing disk information 114
  - marking disk as hot spare 167
  - removing disk for replacement 172
- vxdisksetup 89
- vxdiskunsetup 73
- VxDMP 182
- vxdkpadmin
  - description of 133
  - listing DMP daemons 134
  - options 134
- vxdkpinq 68
- vxdkp
  - backing up using 163
  - described 163
- vxedit
  - excluding disk from hot relocation 167
  - marking disk as hot spare 167
  - removing volumes 151
  - renaming volumes 150
  - setting volumes as subvolumes 148
- vxedquota 176
- vxevac 169
- VxFS 214
  - inode 215
  - inode size 214



- inode structure 214
- VxFS commands 305
- VxFS kernel extensions 187
- vxinfo 129
- vxinstall 133
  - excluding disks from install 57
  - initializing volume manager 54
  - installation option 46
- VxIO 182
- vxiod 69, 81
- vxkextadm 187
- vxlicinst 53, 187
- vxlicrep 54
- vxmake 98
  - creating layered volumes 147
  - creating plexes 137
  - creating subdisks 137
  - creating volumes 137
  - description file 139
- vxnotify 69, 81, 129
- vxplex
  - removing a mirror from a volume 142
- VxPortal 182
- vxprint 83, 110, 118, 125
  - viewing snapshots 158
- VxQIO 182
- vxquot 177
- vxquota 177
- vxquotaoff 178
- vxquotaon 176
- vxreattach 174
- vxrecover 136
- vxrelayout 150
- vxrelocd 69, 80–81, 129, 167
- vxresize 125–126, 185
- vxrestore
  - restored 163
  - restoring files interactively 165
  - using to restore file systems 164
- vxsparecheck 80, 129
- VxSPEC 182
- vxsvc 80–81, 83
- vxtask 136
  - monitoring disk evacuation 169
  - monitoring plex attachments 141
  - monitoring volume relayout 150
- VxVM commands 300
- VxVM tunables 183
- VxVM volume 120
- vxvm-recover 80
- vxvm-startup2 80
- vxvm-sysboot 80
- vxvol 83, 137
  - stopping a volume 151
  - stopping all volumes in a disk group 154

## W

- Web VCS interface 327

## X

- X.25 and SNA 331





## Introducing VERITAS Foundation Suite for AIX







# Introducing VERITAS Foundation Suite for AIX



**Redbooks**

**Helping system administrators familiar with VERITAS Foundation Suite on other UNIX platforms**

**Comparisons with VERITAS Foundation Suite for Sun Solaris as well as AIX's native LVM and JFS/JFS2**

**Providing hints and tips to migrate the existing environment to the AIX environment**

VERITAS Software products, such as VERITAS Volume Manager (VxVM) and VERITAS File System (VxFS), are popular on other UNIX platforms, and VERITAS announced these products for AIX in the early of May 2002. VERITAS Foundation Suite for AIX includes VERITAS Volume Manager for AIX and VERITAS File System for AIX. These products will be useful for people who are familiar with the same products on other UNIX platforms since VERITAS Foundation Suite for AIX provides a similar environment for them. That means they can use their same skills and knowledge in the AIX environment.

This IBM Redbook will not only help system administrators who are familiar with VERITAS Foundation Suite on other UNIX platforms but also AIX system administrators who are familiar with AIX's native Logical Volume Manager (LVM) and Journaled File Systems (JFS) and would like to learn more about VERITAS Foundation Suite for AIX.

This redbook will compare VERITAS Foundation Suite for AIX with VERITAS Foundation Suite for Sun Solaris as well as AIX's native LVM and JFS/JFS2. One of the main focuses is the procedures, hints, and tips to migrate the existing environment to the AIX environment. It will also cover all the details about the functionalities of these products, including planning, installation, configuration, administration, performance, tuning, and troubleshooting. This redbook also introduces VERITAS Cluster Server in Appendix D for those interested in setting up a clustered environment.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)