DMF™ 6 Administrator Guide

007–5484–016

COPYRIGHT
© 2008-2015 Silicon Graphics International Corp. All Rights Reserved; provided portions may be copyright in third parties, as indicated elsewhere herein. No permission is granted to copy, distribute, or create derivative works from the contents of this electronic documentation in any manner, in whole or in part, without the prior written permission of SGI.

LIMITED RIGHTS LEGEND
The software described in this document is "commercial computer software" provided with restricted rights (except as to included open/free source) as specified in the FAR 52.227-19 and/or the DFAR 227.7202, or successive sections. Use beyond license provisions is a violation of worldwide intellectual property laws, treaties and conventions. This document is provided with limited rights as defined in 52.227-14.

TRADEMARKS AND ATTRIBUTIONS
CXFS, DMF, IRIX, SGI, SGI InfiniteStorage, the SGI logo, Silicon Graphics, and XFS are trademarks or registered trademarks of Silicon Graphics International Corp. or its subsidiaries in the United States and other countries.

AMPEX is a trademark of Ampex Corporation. DLT is a trademark of Quantum Corporation. GNOME is a trademark of the GNOME Foundation. Firefox and the Firefox logo are registered trademarks of the Mozilla Foundation. HP is a trademark of Hewlett-Packard Company. IBM and MVS are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Intel and Itanium are trademarks or registered trademarks of Intel Corporation in the United States and other countries. Lustre is a trademark and Oracle and Java are registered trademarks of Oracle and/or its affiliates. Internet Explorer, Microsoft, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Linux is a registered trademark of Linus Torvalds in the U.S. and other countries. MIPSpro is a trademark of MIPS Technologies, Inc., used under license by Silicon Graphics, Inc., in the United States and/or other countries worldwide. Red Hat and Red Hat Enterprise Linux are registered trademarks of Red Hat, Inc. in the United States and other countries. Solaris, and Sun are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the United States and other countries. SLES, SUSE, and YAST are registered trademarks of SUSE LLC in the United States and other countries. UNIX is a registered trademark of the Open Group in the United States and other countries. All other trademarks mentioned herein are the property of their respective owners.

New Features in this Guide

This revision supports DMF 6.5 and includes the following changes:

- Configuration file changes:
  - New parameter in the `drivegroup` object:
    - `REDIRECT_TIME_DG_DOWN` specifies the number of seconds that DMF will wait before redirecting recalls; after all drives have been down for *REDIRECT_TIME_DG_DOWN* seconds, DMF will start forwarding every recall to the VG that holds the secondary copy.

    New parameter in the `filesystem` object:
    - `RELEASETHREADS` specifies the number of threads in the DMF daemon that will be created to handle requests to release filesystem blocks when files are made offline (`OFL`).
  - New parameters in the `taskgroup` object:
    - `DMMIGRATE_MIN_OFFLINE_AGE` specifies the value used for the `-p` option (in minutes) when `dmmigrate`(8) is executed by the `run_dmmigrate.sh` task. This value is a file's minimum age in order for its data blocks to be released, making the file offline.
    - `DMMIGRATE_MIN_OFFLINE_SIZE` specifies the value used for the `-s` option (in bytes) when `dmmigrate` is executed by the `run_dmmigrate.sh` task. This value is a file's minimum size in order for its data blocks to be released, making the file offline.
    - `DMMIGRATE_QUEUE_MAX` specifies the maximum number of `dmmigrate` requests that can be outstanding in the DMF daemon queue at any given time when `dmmigrate` is executed by the `run_dmmigrate.sh` task; when this threshold is reached, the command waits before issuing more requests.

  See:
  - "`drivegroup` Object Parameters" on page 397
  - "`filesystem` Object Parameters" on page 351
  - "`taskgroup` Object Parameters" on page 312

  - `dmmigrate`(8) man page

- Temporary files that are produced by the integrated-backup method now have a project ID of `4136`. To exclude these temporary files from migration, you should add an initial `SELECT_VG` or `SELECT_MSP` parameter that is defined to exclude files with this project ID. This is required under the following conditions:

  - If `DUMP_WORKSPACE` is not set

  - If `DUMP_WORKSPACE` is explicitly set to a location inside the backup filesystem (the filesystem that has `USE_AS_DUMP_DEST` set to `ON`)

  See "`policy` Examples for Integrated Backups" on page 389.

- When DMF is stopped or started using the recommended `service`(8) commands (`service dmf stop` and `service dmf start`), all of the related services are now also stopped or started, irrespective of the status of DMF Manager (the `dmfman` service).

- When configuring the `ident` service, you should use the `only_from` option in the RHEL `/etc/xinetd.d/pidentd` file or SLES `/etc/xinetd.d/auth` file to specify that the service is only run on the DMF server. See step 5 of "Overview of the Installation and Configuration Steps" on page 165.

- "Ensure Quick Communication Among Nodes in the DMF Environment" on page 91:

  - "Preferred Method: Consistent `/etc/hosts` File on All Nodes and Access Local First" on page 91

  - "Acceptable Method: Reliable DNS in a Dedicated HA cluster" on page 93

- "Explicitly Specify the Server on OpenVault Commands" on page 122

Record of Revision

Version	Description
001	December 2008 Original publication. Supports DMF 4.0 in SGI® InfiniteStorage Software Platform (ISSP) 1.5.
002	March 2009 Supports DMF 4.1 in ISSP 1.6.
003	June 2009 Supports DMF 4.2 in ISSP 1.7.
004	September 2009 Supports DMF 4.3 in ISSP 1.8.
005	January 2010 Supports DMF 5.0 in ISSP 2.0.
006	June 2010 Supports DMF 5.1 in ISSP 2.1.
007	September 2010 Supports DMF 5.2 in ISSP 2.2.
008	January 2011 Supports DMF 5.3 in ISSP 2.3.
009	October 2011 Supports DMF 5.5 in ISSP 2.5.
010	April 2012 Supports DMF 5.6 in ISSP 2.6.
011	April 2013 Supports DMF 6.0 in ISSP 3.0.
012	November 2013 Supports DMF 6.1 in ISSP 3.1.

013	May 2014 Supports DMF 6.2 in ISSP 3.2.
014	October 2014 Supports DMF 6.3 in ISSP 3.3.
015	July 2015 Supports DMF 6.4 in ISSP 3.4.
016	November 2015 Supports DMF 6.5 in ISSP 3.5.

Contents

About This Guide . . . . . **li**
Related Publications li
Man Pages lii
User Commands lii
File Formats lii
Administrator Commands lii
Obtaining Publications liii
Conventions liii
Reader Comments liv

1. Introduction **1**
DMF Features 1
Automatic Monitoring of Filesystem Space 2
Easy and Constant Availability of Data 5
Partial-State Files 5
Safety and Scalability 6
Site-Defined Migration Policies 7
A Variety of Migration Targets 7
Support for Fileserving Applications 8
DMF Manager Web Interface 9
DMF Control from Client Platforms 12
Overview of the Windows Client 12
Overview of the IRIX, Mac OS X, Linux, and Solaris Clients 12
High Availability 13
SOAP Web Service 13

Direct Archiving . . . 13
Mounting Services . . . 13
Out-of-Library Tapes . . . 14
How DMF Software Works . . . 14
File States . . . 14
DMF Methods . . . 15
Multiple Storage Tiers . . . 19
Two Tiers . . . 19
Three Tiers using DCM MSP . . . 21
Three Tiers using Fast-Mount Cache . . . 24
Migration Process . . . 27
Recall of File Data . . . 27
Recall from the Appropriate Location . . . 27
Order of Recall . . . 28
Freeing Data Blocks to Make Space for Recalls . . . 30
Fast-Mount Cache Configuration Overview . . . 30
Cache and Secondary-Storage Targets . . . 30
How Fast-Mount Cache Differs from a DCM MSP . . . 31
Fast-Mount Cache Implementation . . . 32
Appropriate Use of Fast-Mount Cache . . . 32
DMF Server Functions . . . 32
Parallel DMF Overview . . . 33
DMF Databases . . . 37
Ensuring Data Integrity . . . 38
DMF Architecture . . . 38
Migrate Groups . . . 42
DMF Capacity . . . 42
Requirements . . . 43

DMF Administrative and Store Directories 44
PostgreSQL Database Server Requirements 47
Server Node Requirements 48
Parallel Data-Mover Node Requirements 48
Mounting Service Requirements 48
License Requirements 48
DMAPI Requirement 48
SAN Switch Zoning or Separate SAN Fabric Requirement 49
DMF Manager Requirements 49
DMF SOAP Requirements 50
DMF Direct Archiving Requirements 50
Fast-Mount Cache Requirements 50
Cloud Storage Requirements 51
Mediaflux Environment Requirements 51
Administration Tasks 51
Initial Planning 52
Installation and Configuration 52
Recurring Administrative Duties 53
Free-Space Management 53
File Ranking 53
Offline Data Management 54
Data Integrity and Reliability 54
Run Backups 55
Audit the Databases and Log files 55
Protect Databases from Loss 55
Remove Old Logs and Journals 56
Hard-Delete Database Entries 56

Commands Overview 56
User Commands 57
Licensing Commands 58
Configuration Commands 58
DMF Daemon and Related Commands 59
Space Management Commands 62
LS Commands 63
DCM MSP Commands 63
Disk MSP Command 64
Other Commands 64

2. DMF Licensing 67

DMF License Types 67
Anticipating Your DMF Data Capacity Requirements 69
Displaying Current DMF Data Capacity Use 72
Parallel Data-Mover Option and Licensing 74
Mounting Services and Licensing 74
Gathering the Host Information 74
Obtaining the License Keys 75
Installing the License Keys 75
Verifying the License Keys 75
DMF Manager Licenses Panel 76
`dmflicense` 77
`lk_verify` 77
For More Information About Licensing 79

3. Best Practices for DMF 81

Best Practices for Installation, Upgrade, and Downgrade 81
Use the Correct Mix of Software Releases 81

Install DMAPI 82
Do Not Use YAST to Configure Network Services 83
Upgrade Nodes in the Correct Order 83
Take Appropriate Steps when Upgrading DMF 84
Contact SGI Support to Downgrade After Using OpenVault™ 4.0 or Later 87
Best Practices for Configuration 87
Plan Ahead for Disaster Recovery 88
Best Practices to Permit Future Disaster Recovery 89
When is Disaster Recovery Possible? 89
What Must Be Protected 89
Why Backups Are Needed 90
Recovery Possibilities 91
Best Practices for Performing Disaster Recovery 91
Ensure Quick Communication Among Nodes in the DMF Environment 91
Preferred Method: Consistent `/etc/hosts` File on All Nodes and Access Local First . 91
Acceptable Method: Reliable DNS in a Dedicated HA cluster 93
Enable RSCN Suppression Appropriately 93
Follow All DMF Requirements 93
Use Supported Libraries and Tape Drives 94
Use Sufficiently Fast Filesystems 94
Configure Passwordless SSH 94
Configure Filesystems and Directories Appropriately for DMF 94
Do Not Place Administrative and Store Directories in a Managed Filesystem . . . 95
Use Different Physical Devices for *HOME_DIR* and *JOURNAL_DIR* 95
Avoid Root Filesystem Location for a Production System 95
Restrict Administrative and Store Directories to DMF Use 95
Use CXFS Filesystems with the Parallel Data-Mover Option 96

Sizing Guidelines 96
HOME_DIR Size 97
JOURNAL_DIR Size 98
SPOOL_DIR Size 98
TMP_DIR Size 98
MOVE_FS Performance and Size 98
Use the Correct Mount Options 99
Migrate at Least Two Copies of a File 100
Safely Make Changes to the DMF Configuration 100
Make and Mount the Required Filesystems First 101
Use Sample DMF Configuration Files 101
Back Up the DMF Configuration File 102
Stop DMF Before Making Changes 102
Always Validate Your Changes 103
Use Inode-Resident Extended Attributes and 256–byte Inodes 104
Limit Path Segment Extension Records 104
Do Not Change Script Names 104
Configure DMF Appropriately with CXFS™ 104
Improve Drive Performance with an Appropriate VG Zone Size 105
Appropriately Force Small Files to Migrate 107
Add HBA Drivers to the `initrd` Image 107
Set `RECALL_NOTIFICATION_RATE` to `0` if CXFS Range Tokens are Disabled 107
Set the `xinetd` `tcpmux` `instances` Parameter Appropriately 108
Avoid Unintentional File Recall by Filesystem Browsers 108
Configure Appropriately for SGI 400 VTL or COPAN MAID Shelves 109
Use Migrate Groups Appropriately 111
Use Fast-Mount Cache Appropriately 112

Use Cloud Storage Appropriately 114
Migrate a Second Copy Outside of the Cloud 114
Use a Firewall 115
Whitelist Only the DMF Server in a Private Cloud 115
Use a Dedicated S3 Bucket 115
Improve Throughput for S3 115
Minimize Access Times by Using NAME_FORMAT 116
Ensure that the Cache Copy is Recalled First 116
Use a Task Group to Run dmmigrate Periodically 116
Restrict the Size of the Alerts and Performance Records Databases 117
Prevent Stalled-Recovery Timeout in a Non-HA Environment 118
Use Appropriate Tape Barcodes 119
Use dmarchive to Copy Archive File Data to Secondary Storage 119
Use an Appropriate Filesystem for a Disk MSP 121
Use Corresponding Drive-Group Names in OpenVault and DMF 121
Use a Private Network Interface in a Parallel Environment 121
Modify Partial-State Capability with Care 122
Use the Correct Child-Process Constraints for a DCM MSP 122
Explicitly Specify the Server on OpenVault Commands 122
Best Practices for Administration 123
Stop and Start DMF Using the service Command 124
Use a Time Synchronization Application 124
Monitor DMF Daily 125
Run Certain Commands Only on a Copy of the DMF Databases 125
Be Aware of Differences in an HA Environment 125
Start Site-Specific Configuration Parameters and Stanzas with “LOCAL_” 125
Use TMF Tracing 126

Run `dmcollect` If You Suspect a Problem . . . 126
Modify Settings If Providing File Access via Samba . . . 127
Export from the Active CXFS Metadata Server . . . 128
Disable Journaling When Loading an Empty Database . . . 128
Use Sufficient Network Bandwidth for Socket Merges . . . 128
Temporarily Disable Components Before Maintenance . . . 128
Gracefully Stop the COPAN Disks and JBFS Disks . . . 129
Reload STK ACSLS Cartridges Properly . . . 129
Disable Zone Reclaim to Avoid System Stalls . . . 129
Set Volume Size If You Want to Use Capacity Features . . . 129
Monitor the Size of the PCP Metrics Archive . . . 130
Be Aware that API Commands Change Without Notice . . . 130
Be Aware of Memory-Mapping Issues . . . 131
Use a Task to Perform Hard-Deletes Periodically . . . 131
Enable the Enhanced-NFS RPC Corruption Workaround Parameter if Needed . . . 131
Use the Appropriate Tool to Load Volumes to an Existing Environment . . . 132
Configure Fibre Channel Switches and Zones Appropriately . . . 132
Ensure that You Follow the Switch Requirements . . . 133
Segregate Tape and Disk HBAs . . . 133
Suppress Change Notification for Switch Ports Connected to Nodes . . . 133
Use `N-port` Topology for LSI Ports Used with Tape Drives . . . 133
Avoid Bottlenecks when Tape Drives and Host Port Speeds Do Not Match . . . 134
Use `dmmove`(8) Safely . . . 136
RHEL: Disable SELinux or Increase the Inode Size . . . 136
Apply Appropriate PostgreSQL Updates . . . 136
Best Practices for Backups . . . 137
Back Up Migrated Filesystems and DMF Databases . . . 137

Migrate All Files Before Performing a Backup . . . 137
Use the Integrated Backup Method . . . 138
Use a Remote Location for the Disaster-Recovery Bundle . . . 138
Retain Log and Journal Files Between Full Backups . . . 138
Best Practices for Optional Tasks . . . 138
Balance Data Among Libraries . . . 139
Prevent Recalls From Waiting for a Busy Volume . . . 140
Use the Immutable Bit if You Want to Prevent File Modification . . . 140

4. Backups and DMF . . . 141

DMF Requirement for Backups . . . 141
Dump Scripts . . . 141
`run_full_dump.sh` . . . 142
`run_partial_dump.sh` . . . 143
Site-Specific Factors to Consider . . . 144
Media-Specific Requirements . . . 144
Backups using Physical Tapes and SGI 400 VTL Tapes . . . 144
Backups using COPAN MAID . . . 145
Considerations for Optimizing Backups . . . 145
Migrate All Files First . . . 146
Consistent Backup Completing Times via `DUMP_MAX_FILESPACE` . . . 147
`DUMP_MAX_FILESPACE` Requires `UPDATE_CTIME` for Incremental Backups . . . 148
Integrated-Backup Method . . . 148
Overview of Integrated Backup . . . 149
Contents of the Integrated Backup . . . 149
Size of the Backup Filesystems . . . 153
DR Bundle Locations . . . 154

Configuring Integrated Backups 154
Precedence of Objects for Integrated Backups 154
Procedure to Configure Objects for Integrated Backups 156
Converting to Integrated Backups 157
Consideration for Integrated Backups: Restored Database May Point to Freed Space . . 160
Monitoring Backup Reports for Integrated Backups 160
Other Backup Methods 161
Report Monitoring for Nonintegrated Tape Backups 161
Storage Used by an FTP MSP or a Standard Disk MSP for Nonintegrated Backups . . 161
Filesystems Used by a DCM MSP for Nonintegrated Backups 161
DMF's Private Filesystems and Nonintegrated Backups 163

5. Installing and Configuring the DMF Environment 165

Overview of the Installation and Configuration Steps 165
Installation and Configuration Considerations 169
ISSP DMF Software 169
Mounting Service Considerations 170
Inode Size Configuration 171
Daemon Database Record Length 173
Interprocess Communication Parameters 175
Automated Maintenance Tasks 175
Networking Considerations for Parallel Data-Mover Option 176
Passwordless SSH Configuration for DMF 176
Suppressing RSCN on Host Port HBAs 177
QLogic® Fibre Channel Switch 178
IPv6 181
Starting and Stopping the DMF Environment 181

Automatic Start After Reboot . . . 182
Preventing Automatic Start After Reboot . . . 182
Explicit Start . . . 183
Explicit Stop . . . 183
Using Out-of-Library Tapes . . . 184
TMF and Out-of-Library Tapes . . . 185
OpenVault and Out-of-Library Tapes . . . 185
Customizing DMF . . . 186
File Tagging . . . 186
Site-Defined Policies . . . 187
Site-Defined Client Port Assignment in a Secure Environment . . . 188
Importing Data From Other HSMs . . . 189

6. DMF Manager . . . 191

Accessing DMF Manager . . . 192
Getting Started with DMF Manager . . . 192
Running Observer Mode or `admin` Mode . . . 195
Observer Mode Functionality . . . 195
`admin` Mode Functionality . . . 196
Admin Mode Access . . . 198
Getting More Information in DMF Manager . . . 198
Setting Panel Preferences . . . 201
Refreshing the View . . . 202
Managing Licenses and Data Capacity with DMF Manager . . . 203
Adding New Licenses . . . 203
Deleting Existing Licenses . . . 204
Viewing the Installed Licenses . . . 205

Showing Current DMF Usage and Licensed Capacity . . . 205
Showing Remaining Storage Capacity . . . 206
Configuring DMF with DMF Manager . . . 210
Limitations to the DMF Configuration Capability . . . 211
Showing All Configured DMF Objects . . . 211
Setting Up a New DMF Configuration File . . . 212
Copying an Object . . . 215
Modifying an Object . . . 217
Creating a New Object . . . 217
Deleting an Object . . . 218
Validating Your Changes . . . 218
Saving Your Configuration Changes . . . 218
Exiting the Temporary Configuration without Saving . . . 219
Displaying DMF Configuration File Parameters . . . 219
Starting and Stopping DMF and the Mounting Service . . . 220
Discovering DMF Problems . . . 221
Filtering Alerts . . . 225
Seeing Relationships Among DMF Components . . . 227
Managing Volumes . . . 229
Managing Libraries . . . 232
Displaying DMF Manager Tasks . . . 233
Monitoring DMF Performance Statistics . . . 233
Using the Statistics Panels . . . 234
Metrics Collection . . . 235
DMF Activity . . . 235
Overview of DMF Activity Reports . . . 235

Key to DMF Activity Reports 236
Example of DMF Activity Report 237
DMF Resources 238
Programs that Update the DMF Resources Reports 239
Filesystem Folder 239
Libraries Report 242
Drive Group Folder 242
Volume Group Folder 244
DCM MSP Folder 246
DMF I/O 248
Displaying Node Status 252
Restoring Filesystems in DMF Manager 255
Restore a Filesystem Component (Integrated Backups) 255
Example: Restoring to the Original Location 258
Example: Restoring to a New Location 260
Restore a Filesystem Component (Nonintegrated Backups) 260
Stop a Restoration Process 261
View Restoration Logs 262
Disaster Recovery in DMF Manager for Basic DMF 262
Viewing the DMF Queue 263
LTFS in DMF Manager 263
Showing LTFS Information in DMF Manager 263
Showing LTFS Progress in DMF Manager 264
Configuring LTFS using DMF Manager 265
Mounting an Unassigned Tape for LTFS Use in DMF Manager 267
Unmounting an LTFS Tape in DMF Manager 269

7. DMF Configuration File 271

Configuration File Overview 272

Object Definitions 272

Stanza Format 274

Units of Measure 275

Device Block-Size Defaults and Bandwidth 276

`base` Object 277

`base` Object Name 277

`base` Object Summary 278

`base` Object Parameters 278

`base` Object Examples 286

`base` Object for Basic DMF 287

`base` Object for Parallel DMF 287

`base` Object for Parallel DMF in an HA Cluster 288

`dmdaemon` Object 289

`dmdaemon` Object Name 289

`dmdaemon` Object Summary 289

`dmdaemon` Object Parameters 290

`dmdaemon` Object Example 293

`node` Object 294

`node` Object Name 294

`node` Object Summary 294

`node` Object Parameters 294

`node` Object Examples 297

`node` Objects for Parallel DMF 297

`node` Objects for Parallel DMF in an HA Cluster 298

`services` Object 299

services Object Name 299
services Object Summary 299
services Object Parameters 300
services Object Examples 301
services object for Parallel DMF 301
services Object for Parallel DMF in an HA Cluster 303
taskgroup Object 304
Overview of the Tasks 304
Backup Tasks 309
taskgroup Object Name 310
taskgroup Object Summary 310
taskgroup Object Parameters 312
taskgroup Object Examples 332
taskgroup Object Example for Integrated Backups 332
taskgroup Object Example for Tape Backups 333
taskgroup Object Example for Disk Backups 335
taskgroup Object Example for Disk Backups Using NFS 335
taskgroup Object Example for Daemon Tasks 336
taskgroup Object Example for Node Tasks 339
taskgroup Object Example for Fast-Mount Cache Tasks 339
taskgroup Object Example for Fast-Mount Cache Tasks Using File Retention . . . 340
taskgroup Object Example for Periodic dmmigrate Tasks 341
taskgroup Object Example for Removing Alerts and Performance Records . . . 341
dumpgroup Object 342
dumpgroup Object Name 342
dumpgroup Object Summary 343

`dumpgroup` Object Parameters . . . 343
`dumpgroup` Object Example for Integrated Backups . . . 345
`device` Object . . . 347
`device` Object Name . . . 348
`device` Object Summary . . . 348
`device` Object Parameters . . . 348
`filesystem` Object . . . 350
`filesystem` Object Name . . . 350
`filesystem` Object Summary . . . 351
`filesystem` Object Parameters . . . 351
`filesystem` Object Examples . . . 360
`filesystem` Object Example for a Managed Filesystem . . . 361
`filesystem` Object Example for Integrated Backups . . . 362
`filesystem` Object Example for Incremental Backups that Restrict File Size . . . 362
`filesystem` Object for DMF Direct Archiving . . . 363
`policy` Object . . . 364
Functions of `policy` Parameters . . . 365
Automated Space-Management Overview . . . 365
File Weighting Overview . . . 365
MSP/VG Selection Overview . . . 366
Rules for `policy` Parameters . . . 366
Managed Filesystem Rules . . . 366
DCM MSP `STORE_DIRECTORY` Rules . . . 367
`policy` Object Name . . . 368
Managed Filesystem `policy` Parameters . . . 368
`policy` Object Summary for a Managed Filesystem . . . 368
Automated Space-Management Parameters for a Managed Filesystem . . . 369

File Weighting Parameters for a Managed Filesystem . . . 371
MSP/VG Selection Parameters for a Managed Filesystem . . . 374
DCM MSP STORE_DIRECTORY policy Parameters . . . 375
policy Object Summary for a DCM MSP . . . 376
Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY . . . 376
File Weighting Parameters for a DCM MSP STORE_DIRECTORY . . . 378
VG Selection Parameters for a DCM MSP STORE_DIRECTORY . . . 380
when Clause . . . 380
ranges Clause . . . 383
policy Configuration Examples . . . 385
Automated Space-Management Example . . . 386
Automated Space-Management Using Ranges Example . . . 387
MSP/VG Selection Example . . . 388
policy Examples for Integrated Backups . . . 389
fastmountcache Object . . . 390
fastmountcache Object Name . . . 390
fastmountcache Object Summary . . . 390
fastmountcache Object Parameters . . . 390
fastmountcache Object Examples . . . 391
fastmountcache with an MG . . . 391
fastmountcache with a Mix of Members . . . 391
Objects Related to an LS . . . 392
libraryserver Object . . . 392
libraryserver Object Name . . . 393
libraryserver Object Summary . . . 393
libraryserver Object Parameters . . . 393
drivegroup Object . . . 395

drivegroup Object Name 396
drivegroup Object Summary 396
drivegroup Object Parameters 397
volumegroup Object 410
volumegroup Object Name 410
volumegroup Object Summary 410
volumegroup Object Parameters 411
volumegroup Object Example with an AG 423
volumegroup Object Example Overriding drivegroup Settings 426
migrategroup Object 426
migrategroup Object Name 427
migrategroup Object Summary 427
migrategroup Object Parameters 427
migrategroup Object Example with Multiple MGs 431
Single migrategroup Object Example Using the ROUND_ROBIN_BY_BYTES Strategy 431
migrategroup Object Example Using the ROUND_ROBIN_BY_FILES Strategy . . 431
resourcescheduler Object 432
resourcescheduler Object Name 432
resourcescheduler Object Summary 432
resourcescheduler Object Parameters 432
resourcewatcher Object 433
resourcewatcher Object Name 434
resourcewatcher Object Summary 434
resourcewatcher Object Parameters 434
allocationgroup Object 434
allocationgroup Object Name 435

`allocationgroup` Object Summary 435

`allocationgroup` Object Parameters 435

Examples of Configuring an LS 436

Example: LS with a Resource Watcher, Two DGs, and an AG 436

Example: LS for Fast-Mount Cache 439

LS Tasks 441

Overview of LS Tasks 442

LS `taskgroup` Object with One VG 443

LS `taskgroup` Object with Multiple VGs 444

LS Database Records 444

`msp` Objects 446

`msp` Object Name 446

`NAME_FORMAT` Details 446

`msp` Object for a Cloud MSP 448

`msp` Object Summary for a Cloud MSP 449

`msp` Object Parameters for a Cloud MSP 449

`msp` Object Example for an S3 Cloud MSP 453

`msp` Object Example for a Scality Cloud MSP 454

`msp` Object for an FTP MSP 455

`msp` Object Summary for an FTP MSP 455

`msp` Object Parameters for an FTP MSP 455

`msp` Object Example for an FTP MSP 458

`msp` Object for a DCM MSP 459

`msp` Object Summary for a DCM MSP 460

`msp` Object Parameters for a DCM MSP 460

`msp` Object Example for a DCM MSP 465

`msp` Object for a Disk MSP 466

msp Object Summary for a Disk MSP 467
msp Object Parameters for a Disk MSP 467
msp Object Example for a Disk MSP 470
Summary of the Configuration-File Parameters 472

8. Parallel DMF Configuration 485
Parallel DMF Configuration Procedure 485
Determining the State of Parallel Data-Mover Nodes 488
Disabling Parallel Data-Mover Nodes 489
Reenabling Parallel Data-Mover Nodes 489

9. Mounting Service Configuration Tasks 491
OpenVault Configuration Tasks 491
Configure the OpenVault Server 492
Configure OpenVault for DMF Use 494
Configure OpenVault for Each Parallel Data-Mover Node 498
Configure OpenVault on the DMF Server If on a Different Host 502
Configure OpenVault for a Drive Group 502
TMF Configuration Tasks 505

10. Message Log Files 507

11. Automated Space Management 509
The dmfsmon Daemon and dmfsfree Command 509
Generating the Candidate List 510
Selection of Migration Candidates 511
Space Management and the DCM MSP 513
Automated Space Management Log File 513

12. The DMF Daemon 515
Daemon Processing 515
Daemon Database and dmdadm 517
Overview of the Daemon Database and dmdadm 517
dmdadm Directives 518
dmdadm Field and Format Keywords 520
dmdadm Text Field Order 524
Daemon Logs and Journals 525

13. The DMF Lock Manager 527
dmlockmgr Communication and Log Files 527
dmlockmgr Individual Transaction Log Files 529

14. Library Servers and Media-Specific Processes 531
LS Operations 532
LS Overview 532
LS Directories 533
Media Concepts 534
CAT Records 536
VOL Records 537
LS Journals 538
LS Logs 539
Volume Merging 543
dmcatadm Command 544
dmcatadm Directives 545
dmcatadm Keywords 548
dmcatadm Text Field Order 553
dmvoladm Command 554
dmvoladm Overview 554

dmvoladm Directives 555
dmvoladm Field Keywords 557
dmvoladm Text Field Order 563
dmvoladm Examples 564
dmatread Command 567
dmatsnf Command 568
dmaudit verifymsp Command 568
Cloud MSP 569
Cloud MSP Overview 569
Cloud MSP Processing of Requests 569
Cloud MSP Activity Log 571
Cloud MSP Messages 571
Cloud MSP Limitations and Considerations 572
FTP MSP 573
FTP MSP Overview 573
FTP MSP Processing of Requests 573
FTP MSP Activity Log 575
FTP MSP Messages 576
FTP MSP Limitations and Considerations 576
Disk MSP 576
Disk MSP Overview 577
Disk MSP Processing of Requests 577
Disk MSP Activity Log 578
DCM MSP 579
dmdskvfy Command 579
Moving Migrated Data 579
LS Error Analysis and Avoidance 580
LS Drive Scheduling 582

LS Status Monitoring 582

15. DMF Maintenance 585

Retaining Old DMF Daemon Log Files 585

Retaining Old DMF Daemon Journal Files 586

Cleaning Up Obsolete Database Entries 586

Using `dmfill` 587

Viewing Drive Statistics 587

Temporarily Disabling Components 589

Disable an OpenVault DCP 590

Disable an OpenVault Drive 591

Disable an OpenVault Library 592

Disable a TMF Drive 594

Stop the COPAN VTL 595

16. DMF SOAP Server 597

Overview of DMF SOAP 597

Accessing the DMF SOAP and WSDL 599

Starting and Stopping the DMF SOAP Service 600

Starting the `dmfsoap` Service 601

Preventing Automatic Start of `dmfsoap` After Reboot 601

Explicitly Stopping `dmfsoap` 601

Security / Authentication 601

DMF SOAP Sample Client Files 601

17. DMF Queue Viewing 605

QView Tab and the `dmqview` Command 605

Taking a Snapshot of the DMF Requests 606

Queue Reports 606
Summary Report and Corresponding State Reports 606
Composite Reports 609
Canceling a Request 613
Reprioritizing a Request 615
Modifying the Report Output 615
Modifying the QView Display 615
Modifying the `dmqview` Output 617
Example of Queue-Related Tasks 619
Database Tables and Fields 620
PostgreSQL Database Tables 621
Fields for the Chunk Database Table 624
Fields for the Connect Database Table 625
Fields for the Daemon Request Database Table 626
Fields for the Filesys Database Table 627
Fields for the Mover Database Table 628
Fields for the MSP Request Database Table 629

18. SGI LTFS 631
SGI LTFS Features 631
SGI LTFS Requirements 632
Best Practices for SGI LTFS 632
Avoid Parallel Use 632
Verify Copies are Present Before Deleting Originals 632
Ensure that Files are Online Before Copying to LTFS 633
SGI LTFS Configuration 633
`sgi-ltfs.conf` File Format 633
`sgi-ltfs.conf` Parameters 634

`sgi-ltfs.conf` Examples 635
Borrowing Drives (Simple) 635
Borrowing Drives (More Complex) 636
Using the LTFS Drive 636
`sgi-ltfs` Command Examples 637
Reading an LTFS Tape 637
Reserving a Drive and Preparing a New LTFS Tape 638
Reading a T10000C-Formatted Tape on a T10000D Drive 639
Reformatting a T100002 Tape 639

19. Troubleshooting 641

Replacing a Bad DMF Copy with a Good DMF Copy 642
Restoring Files from Backups 643
Overview of Restoring Files 643
Ensuring Accuracy with `xfsdump` 645
Backing Up and Restoring Files without the DMF Scripts 645
Filesystem Consistency with `xfsrestore` 646
Recovery and Restoration of DMF Database 646
Database Backups 647
Database Recovery Procedure 647
Database Restore Procedure for Integrated Backups 650
Backup Consistency Errors 650
Disaster Recovery Procedures 650
Disaster Recovery for Basic DMF without HA 651
Disaster Recovery for Parallel DMF 655
Disaster Recovery for DMF with HA 657
Filesystem Errors 658

Unable to Use the dmi Mount Option 660
EOT Error 660
Tape Drive Not Claimed by ts 660
Drive Entry Does Not Correspond to an Existing Drive (OpenVault) 660
Drive Does Not Exist (TMF) 661
No Usable Drives 661
DMF Manager Errors 666
DMF Statistics are Unavailable Error Message 666
DMF Statistics Graphs are Empty 668
OpenVault Library Is Missing 668
Delay In Accessing Files in an SMB/CIFS Network Share 668
Operations Timeout or Abort on Windows 669
Windows Explorer Hangs 669
Poor Migration Performance 669
Poor Recall Performance 669
Remote Connection Failures 670
yast2 Disk Space Warning 670
Linux CXFS Clients Cannot Mount Managed Filesystems 670
Mount Errors for a Drive 671
Verifying the Integrity of Library Server Volumes 672
dmemptytape Process Did Not Complete 674
DMF Database Server Startup Issues 674
Using SGI Knowledgebase 675
Reporting Problems to SGI 675

Appendix A. Messages 677

dmcatadm Message Interpretation 677

dmvoladm Message Interpretation 679

Appendix B. DMF User Library libdmfusr.so 681

Overview of the Distributed Command Feature and libdmfusr.so 681
Considerations for IRIX® 684
libdmfusr.so Library Versioning 684
libdmfusr.so.2 Data Types 686
DmuAllErrors_t 686
DmuAttr_t 687
DmuByteRange_t 688
DmuByteRanges_t 689
DmuCompletion_t 692
DmuCopyRange_t 693
DmuCopyRanges_t 693
DmuErrHandler_f 694
DmuErrInfo_t 695
DmuError_t 695
DmuEvents_t 696
DmuFhandle_t 696
DmuFsysInfo_t 696
DmuFullRegbuf_t 697
DmuFullRegion_t 698
DmuFullstat_t 698
DmuFullstat_2_t 699
DmuOperandType_t 700
DmuPriority_t 701
DmuRegion_t 701
DmuRegionbuf_t 701
DmuReplyOrder_t 702

DmuReplyType_t 702
DmuSeverity_t 702
DmuVolGroup_t 703
DmuVolGroups_t 703
User-Accessible API Subroutines for libdmfusr.so.2 705
Context-Manipulation Subroutines 705
DmuCreateContext() Subroutine 705
DmuChangedDirectory() Subroutine 707
DmuDestroyContext() Subroutine 707
Filesystem-Information Subroutine 708
DMF File-Request Subroutines 709
Copy File Requests 710
Archive File Requests 712
Fullstat Requests 714
Put File Requests 717
Get File Requests 720
Setprojid File Requests 723
Settag File Requests 725
Request-Completion Subroutines 727
DmuAwaitReplies() Subroutine 728
DmuFullstatCompletion() Subroutine 729
DmuGetNextReply() Subroutine 730
DmuGetThisReply() Subroutine 732

Appendix C. Site-Defined Policy Subroutines and the sitelib.so Library 735

Overview of Site-Defined Policy Subroutines 735
Getting Started with Custom Subroutines 736
Considerations for Writing Custom Subroutines 738

sitelib.so Data Types 739
DmaContext_t 739
DmaFrom_t 740
DmaIdentity_t 740
DmaLogLevel_t 742
DmaRealm_t 742
DmaRecallType_t 742
SiteFncMap_t 743
Site-Defined Policy Subroutines 743
SiteArchiveFile() 743
SiteCreateContext() 745
SiteDestroyContext() 746
SiteKernRecall() 746
SitePutFile() 748
SiteWhen() 750
Helper Subroutines for sitelib.so 752
DmaConfigStanzaExists() 752
DmaGetConfigBool() 753
DmaGetConfigFloat() 754
DmaGetConfigInt() 755
DmaGetConfigList() 756
DmaGetConfigStanza() 757
DmaGetConfigString() 758
DmaGetContextFlags() 759
DmaGetCookie() 759
DmaGetDaemonMigGroups() 760
DmaGetDaemonVolAndMigGroups() 760
DmaGetDaemonVolGroups() 761

DmaGetMigGroupMembers() 761
DmaGetProgramIdentity() 762
DmaGetUserIdentity() 762
DmaSendLogFmtMessage() 763
DmaSendUserFmtMessage() 764
DmaSetCookie() 765

Appendix D. Third-Party Backup Package Configuration 767

Using Third-Party Backup Packages 767
Overview of Using Third-Party Backup Packages 767
do_predump.sh Script 768
DUMP_DATABASE_COPY Configuration File Parameter in the taskgroup Object . . . 768
Hard Deletes and Third-Party Backup 769
Unprivileged Users 769
Procedure to Use a Third-Party Backup Package 769
ASG-Time Navigator 769
taskgroup Object Example for Third-Party Backups 770

Appendix E. Converting from IRIX DMF to Linux® DMF 773

Appendix F. Converting from TMF to OpenVault 777

Appendix G. Considerations for Partial-State Files 779

Performance Cost Due to Lack of Linux Kernel Support 779
Inability to Fulfill Exact Byte Range Requests 780

Appendix H. Case Study: Impact of Zone Size on Tape Performance . . 781

Appendix I. Historical Feature Information 783

End of Life for the Tape Autoloader API with DMF 2.6.3 783
DMF Directory Structure Prior to DMF Release 2.8 783
End of Life for the Tape MSP after DMF 3.0 784
DMF User Library (`libdmfusr.so`) Update in DMF 3.1 784
Downgrading and the Site-Tag Feature Introduced in DMF 3.1 785
Downgrading and the Partial-State File Feature Introduced in DMF 3.2 786
`dmaudit`(8) Changes in DMF 3.2 787
Logfile Changes in DMF 3.2 787
Possible DMF Database Lock Manager Incompatibility On Upgrades as of DMF 3.8.3 . . 788

Appendix J. Using `dmmaint` to Install Licenses and Configure DMF . . 789

Overview of `dmmaint` 789
Installing the DMF License 791
Using `dmmaint` to Define the Configuration File 791

Glossary 793

Index 853

Figures

Figure 1-1 DMF Cycle 2

Figure 1-2 Free-Space Minimum Threshold 4

Figure 1-3 DMF Manager 11

Figure 1-4 DMF Methods: Before Migrating 17

Figure 1-5 DMF Methods: After Migrating Data and Freeing Space 18

Figure 1-6 Two Tiers: Migrating File Data 20

Figure 1-7 Two Tiers: Freeing and Recalling File Data 21

Figure 1-8 DCM MSP: Migrating File Data 22

Figure 1-9 DCM MSP: Freeing and Recalling File Data 23

Figure 1-10 Fast-Mount Cache: Migrating File Data 25

Figure 1-11 Fast-Mount Cache: Freeing and Recalling File Data 26

Figure 1-12 Recall Order 29

Figure 1-13 Basic DMF Product in an NFS Environment 34

Figure 1-14 Basic DMF Product in a CXFS Environment 34

Figure 1-15 Parallel DMF in a CXFS Environment 36

Figure 1-16 Basic DMF Architecture 39

Figure 1-17 LS Architecture 40

Figure 2-1 DMF Licenses 69

Figure 2-2 Data that Counts Towards the Capacity License 71

Figure 2-3 Licenses 76

Figure 3-1 Components Required by the DMF System 90

Figure 3-2 Archiving Files from an Archive Filesystem to Secondary Storage 120

Figure 3-3 DMF Direct Archiving 120

Figure 4-1 DMF Integrated Backups 152
Figure 4-2 Integrated Backup Directory Structures 153
Figure 4-3 Order of Precedence for Dump Parameters 155
Figure 6-1 DMF Manager Overview Panel 194
Figure 6-2 Overview Key to Symbols 199
Figure 6-3 Displaying Information About an Icon 200
Figure 6-4 “What Is ...” Information 201
Figure 6-5 DMF Manager Overview Preferences Panel 202
Figure 6-6 Adding a License Key in DMF Manager 204
Figure 6-7 Data Under Management and Data Charged to the Licenses 206
Figure 6-8 DMF Capacity 207
Figure 6-9 Remaining DMF Capacity 209
Figure 6-10 Temporary Workspace for a Preconfigured DCM MSP Sample 213
Figure 6-11 Naming a Copied Object 216
Figure 6-12 DMF Configuration Parameters in DMF Manager 220
Figure 6-13 DMF Manager Showing Problems in the DMF System 221
Figure 6-14 Alerts Key 222
Figure 6-15 Unfiltered Alerts 223
Figure 6-16 DMF Manager Alerts Panel and Help Information 224
Figure 6-17 Define Filters for Alerts 225
Figure 6-18 Adding Another Filter Rule 226
Figure 6-19 Filtered Alerts 227
Figure 6-20 Relationships Among DMF Components 228
Figure 6-21 DMF Manager Volumes Panel 229
Figure 6-22 Changing Hold Flags in DMF Manager 231
Figure 6-23 DMF Activity 238

Figure 6-24 Filesystem Resource Graph 241
Figure 6-25 Drive Group Resource Information 243
Figure 6-26 Volume Group Resource Graph 245
Figure 6-27 DCM MSP Resource Graph 247
Figure 6-28 DMF I/O Custom Chart Creation 250
Figure 6-29 DMF I/O 251
Figure 6-30 Node State 252
Figure 6-31 Node Details 254
Figure 6-32 Restoring a File from **Available Dumps (Integrated Method)** 256
Figure 6-33 Restoring a Filesystem Component using DMF Manager (Integrated Backups) 259
Figure 6-34 Restoring a Filesystem Component using DMF Manager (Nonintegrated Backups) 261
Figure 6-35 LTFS Information 264
Figure 6-36 LTFS Messages 265
Figure 6-37 Configuring LTFS in DMF Manager 266
Figure 6-38 Mounting an LTFS Tape 268
Figure 6-39 Unmounting an LTFS Tape 270
Figure 7-1 Concepts of Free-Space Minimum and Target 370
Figure 11-1 Relationship of Automated Space Management Targets 512
Figure 14-1 Media Concepts 536
Figure 16-1 DMF SOAP Operation Documentation 598
Figure 16-2 WSDL Definition 600
Figure 17-1 QView: Summary Reports 607
Figure 17-2 Processing Report 612
Figure 17-3 Canceling a Request 613
Figure 17-4 Fields Available for Display by a QView **Active** Report 616
Figure 19-1 Disaster Recovery 652

Tables

Table 1-1 DMF Administrative and Store Directories 45
Table 2-1 Data-Capacity License Amounts 68
Table 3-1 DMF Exclusive Use of Filesystems 96
Table 3-2 Minimum Sizes for DMF Directories 97
Table 3-3 Required and Suggested Mount Options 99
Table 3-4 Tools to Load Volumes to an Existing DMF/OpenVault Environment . . . 132
Table 5-1 Default Maximum File Regions for XFS and CXFS Filesystems 172
Table 6-1 DMF Manager Panel Menus 193
Table 7-1 Automated Maintenance Task Summary 305
Table 7-2 Dump Parameters According to DMF Backup Method 309
Table 7-3 `NAME_FORMAT` Strings 447
Table 7-4 Configuration Parameters 472
Table 10-1 Message Types and Levels 508
Table 13-1 `dmlockmgr` Token Files 528
Table 17-1 Summary Reports 608
Table 17-2 Composite Reports 609
Table 17-3 PostgreSQL Database Tables 622
Table 17-4 Chunk Fields 624
Table 17-5 Connect Fields 625
Table 17-6 Daemon Request Fields 626
Table 17-7 Filesys Fields 627
Table 17-8 Mover Fields 628
Table 17-9 MSP Request Fields 629

Examples

Example 7-1 `base` Object for Basic DMF 287

Example 7-2 `base` Object for Parallel DMF 287

Example 7-3 `base` Object for Parallel DMF in an HA Cluster 288

Example 7-4 `dmdaemon` object 293

Example 7-5 `node` Objects for Parallel DMF 297

Example 7-6 `node` Objects for Parallel DMF in an HA Cluster 298

Example 7-7 `services` object for Parallel DMF 301

Example 7-8 `services` Object for Parallel DMF in an HA Cluster 303

Example 7-9 `taskgroup` Object for Integrated Backups 332

Example 7-10 `taskgroup` Object for Tape Backups 333

Example 7-11 `taskgroup` Object for Disk Backups 335

Example 7-12 `taskgroup` Object for Disk Backups Using NFS 335

Example 7-13 `taskgroup` Object for Daemon Tasks 336

Example 7-14 `taskgroup` Object for Node Tasks with Parallel DMF 339

Example 7-15 `taskgroup` Object for Fast-Mount Cache 339

Example 7-16 `taskgroup` Object for Fast-Mount Cache Using File Retention 340

Example 7-17 `taskgroup` Object for Periodic `dmmigrate` Example 341

Example 7-18 `taskgroup` Object for Removing Alerts and Performance Records . . . 341

Example 7-19 `dumpgroup` Object Example for Integrated Backups 345

Example 7-20 `filesystem` Object for a Managed Filesystem 361

Example 7-21 `filesystem` Object for Integrated Backups 362

Example 7-22 `filesystem` Objects for Incremental Backups that Restrict File Size . . 362

Example 7-23 `filesystem` Object for DMF Direct Archiving 364

Example 7-24 `policy` Object for Automated Space Management 386
Example 7-25 `policy` Object for Automated Space Management Using Ranges . . . 387
Example 7-26 `policy` Object for an MSP/VG 388
Example 7-27 `policy` Objects for Integrated Backups 389
Example 7-28 `fastmountcache` with an MG 391
Example 7-29 `fastmountcache` with a Mix of Members 391
Example 7-30 `volumegroup` example with an AG 424
Example 7-31 `migrategroup` Object with Multiple MGs 431
Example 7-32 Single `migrategroup`Using the `ROUND_ROBIN_BY_BYTES` Strategy . . 431
Example 7-33 `migrategroup` Using the `SEQUENTIAL` Strategy 431
Example 7-34 `libraryserver` Object with a Resource Watcher, Two DGs, and an AG . 436
Example 7-35 `libraryserver` and Associated Objects for Fast-Mount Cache 439
Example 7-36 `taskgroup` Object for LS with One VG 443
Example 7-37 `msp` Object for an S3 Cloud MSP 453
Example 7-38 `msp` Object for a Scality Cloud MSP 454
Example 7-39 `msp` Object for an FTP MSP 458
Example 7-40 Configuration Stanzas Associated with a DCM MSP 465
Example 7-41 `msp` Object for a Disk MSP 470
Example 14-1 LS Statistics Messages 541
Example 14-2 `dmcatadm list` Directive 552
Example 14-3 `dmvoladm update` Directive 564
Example 14-4 `dmvoladm list` Directive to Show Information for Multiple VSNs . . . 564
Example 14-5 `dmvoladm list` Directive to Show Volumes with a Specific Flag . . . 564
Example 14-6 `dmvoladm list` Directive to Customize a List of Fields 565
Example 14-7 `dmvoladm list` Directive to Show Multiple Flags 566
Example 14-8 `dmvoladm list` Directive to Display Volumes Assigned to a VG . . . 567
Example 14-9 Restoring Hard-deleted Files Using `dmatread` 567

Example 19-1 Database Recovery 649
Example D-1 `taskgroup` Object for Third-Party Backups 770
Example E-1 IRIX to Linux Conversion (Single LS) 776

Procedures

Procedure 5-1 Configuring the DMF Environment 165
Procedure 5-2 Configuring the Daemon Database Record Length 173
Procedure 7-1 Creating LS Database Records 445
Procedure 8-1 Configuring Parallel DMF 485
Procedure 9-1 Configuring OpenVault for a Drive Group 502
Procedure 19-1 Recovering the Databases 648
Procedure E-1 Converting from IRIX DMF to Linux DMF 773

About This Guide

This guide tells you how to configure and administer the SGI® DMF tiered-storage virtualization environment.

Related Publications

For information about this release, see the SGI InfiniteStorage™ Software Platform (ISSP) release notes (`README.txt`) and the DMF release notes (`README_DMF.txt`).

The *DMF 6 Filesystem Audit Guide* describes how to solve problems with DMF should you encounter them.

Also see:

- *COPAN MAID for DMF Quick Start Guide*
- *CXFS 7 Administrator Guide for SGI InfiniteStorage*
- *CXFS 7 Client-Only Guide for SGI InfiniteStorage*
- *High Availability Guide for SGI InfiniteStorage*
- *OpenVault Administrator Guide for SGI InfiniteStorage*
- *SGI 400 VTL for DMF Quick Start Guide*
- *TMF 6 Administrator Guide for SGI InfiniteStorage*
- *XVM Volume Manager Administrator Guide*
- Scality RING documentation provided at:

  docs.scality.com

  Note: You must enter your Scality username and password

- Amazon Simple Storage Service (S3) documentation provided at:

  http://aws.amazon.com/s3/

Man Pages

DMF provides man pages for user commands, file formats, and administrator commands.

User Commands

Man pages are available for the following DMF user commands:

dmarchive(1)	dmdu(1)	dmls(1)	dmversion(1)
dmattr(1)	dmfind(1)	dmput(1)	sgi_dmdu(1)
dmcapacity(1)	dmget(1)	dmtag(1)	sgi_dmfind(1)
dmcopy(1)			sgi_dmls(1)

File Formats

Man pages are available for the following DMF file formats:

dmf.conf(5)
trxj(5)

Administrator Commands

Man pages are available for the following DMF administrator commands:

dmatread(8)	dmdbrecover(8)	dmfsfree(8)	dmscanfs(8)
dmatsnf(8)	dmdidle(8)	dmfsmon(8)	dmselect(8)
dmatvfy(8)	dmdskfree(8)	dmhdelete(8)	dmsnap(8)
dmaudit(8)	dmdskvfy(8)	dmlockmgr(8)	dmsort(8)
dmcatadm(8)	dmdstat(8)	dmmigrate(8)	dmstat(8)
dmcheck(8)	dmdstop(8)	dmmove(8)	dmtapestat(8)
dmclripc(8)	dmdump(8)	dmmvtree(8)	dmunput(8)
dmcollect(8)	dmdumpj(8)	dmnode_admin(8)	dmusage(8)
dmconfig(8)	dmemptytape(8)	dmov_keyfile(8)	dmusrcmd(8)
dmcopan(8)	dmfdaemon(8)	dmov_loadtapes(8)	dmvoladm(8)
dmdadm(8)	dmfill(8)	dmov_makecarts(8)	dmxfsprune(8)
dmdate(8)	dmflicense(8)	dmqview(8)	dmxfsrestore(8)
dmdbcheck(8)			sitelibverify(8)

There is also the sgi-ltfs(8) man page.

Obtaining Publications

You can obtain SGI documentation as follows:

- Log in to the SGI Customer Portal at http://support.sgi.com. Click the following:

  Support by Product
  > ***productname***
  > Documentation

  If you do not find what you are looking for, click **Search Knowledgebase**, enter a document-title keyword, select the category **Documentation**, and click **Search**.

- The `/docs` directory on the ISSP DVD or in the online download directory contains the following:
  - The ISSP release note: `/docs/README.txt`
  - DMF release notes: `/docs/README_DMF.txt`
  - A complete list of the packages and their location on the media: `/docs/RPMS.txt`
  - The packages and their respective licenses: `/docs/PACKAGE_LICENSES.txt`
- The release notes and manuals are provided in the `noarch/sgi-isspdocs` RPM and will be installed on the system into the following location:

  `/usr/share/doc/packages/sgi-issp-`*`ISSPVERSION/TITLE`*

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage` (*x*)	Man page section identifiers appear in parentheses after man page names.
variable	Italic typeface denotes variable entries and words or concepts being defined.

`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. (Output is shown in nonbold, fixed-space font.)
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this publication, contact SGI. Be sure to include the title and document number of the publication with your comments. (Online, the document number is located in the front matter of the publication. In printed publications, the document number is located at the bottom of each page.)

You can contact SGI in either of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system:

  http://www.sgi.com/support/supportcenters.html

SGI values your comments and will respond to them promptly.

Introduction

This chapter provides an overview of SGI® DMF tiered-storage virtualization software. It discusses the following:

- "DMF Features" on page 1
- "How DMF Software Works" on page 14
- "Requirements" on page 43
- "Administration Tasks" on page 51

DMF Features

DMF software transparently moves file data from high-performance but expensive disk to levels of decreased-performance but inexpensive media known as *secondary storage*. This lets you cost-effectively maintain a seemingly infinite amount of data without sacrificing accessibility for users.

This section discusses the following features of DMF software:

- "Automatic Monitoring of Filesystem Space" on page 2
- "Easy and Constant Availability of Data" on page 5
- "Partial-State Files" on page 5
- "Safety and Scalability" on page 6
- "Site-Defined Migration Policies" on page 7
- "A Variety of Migration Targets" on page 7
- "Support for Fileserving Applications" on page 8
- "DMF Manager Web Interface" on page 9
- "DMF Control from Client Platforms" on page 12
- "High Availability" on page 13
- "SOAP Web Service" on page 13

- "Direct Archiving" on page 13
- "Mounting Services" on page 13
- "Out-of-Library Tapes" on page 14

Automatic Monitoring of Filesystem Space

A *managed filesystem* is an XFS or CXFS filesystem mounted with the *Data Management Application Programming Interface* (DMAPI) enabled and for which DMF software can migrate and/or recall migrated data. DMF software continuously monitors managed filesystems on high-performance disk so that it can maintain a certain amount of free space in those filesystems. This free space permits the creation of new files and the recall of previously migrated files. Figure 1-1 describes the concept of the DMF migration cycle between the managed filesystem and the secondary storage.

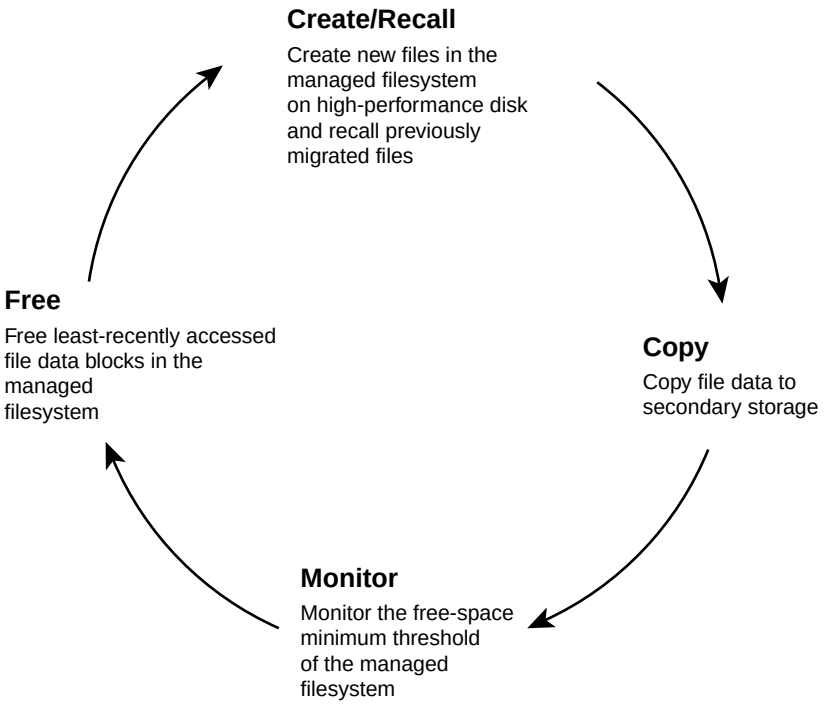

Figure 1-1 DMF Cycle

DMF software automatically detects a drop below the free-space threshold. DMF software then transparently moves file data from the managed filesystem to the secondary storage by freeing the data blocks of files that have already been migrated. File migration occurs in two stages:

- Stage One: A file's data is copied (*migrated*) to secondary storage.
- Stage Two: After the copy is secure, the file is eligible to have its data blocks released. This occurs only after a minimum free-space threshold is reached or when a manual request to free a file's disk blocks is made via the `dmput -r` command. DMF software choses file data to free according to site-defined policies involving size and access time.

For example, Figure 1-2 shows a configuration where DMF software will free the data blocks of less-recently accessed files (such as represented by the letter "**A**") to empty the managed filesystem well below the threshold as new files are added or as previously migrated files (such as represented by the letters "**B**" and "**E**") are recalled. Despite the movement of data, all content is accessible all of the time.

Note: When configured according to best practices, DMF software makes two copies of migrated data for safety reasons. Data will be recalled from a second copy only if necessary. For simplicity, Figure 1-2 does not show the second copy of file data.

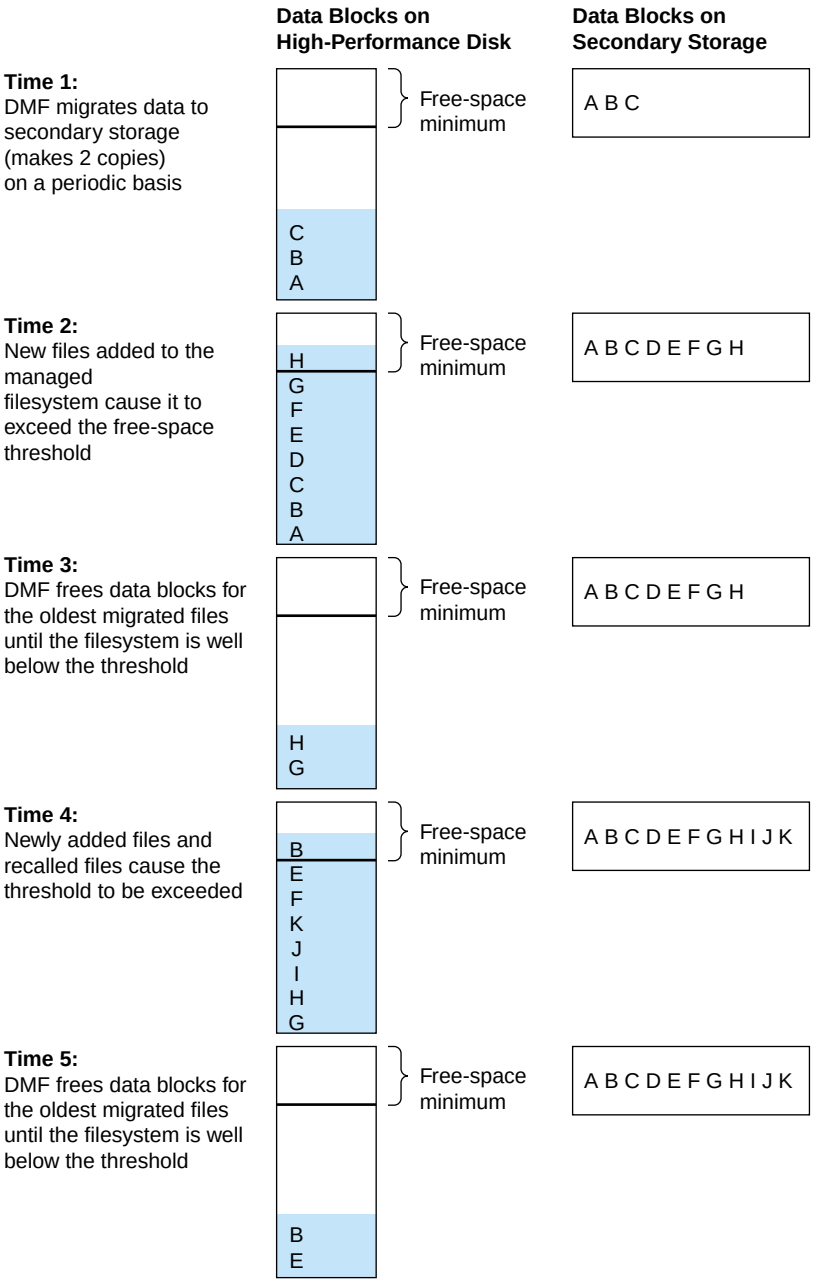

Figure 1-2 Free-Space Minimum Threshold

Easy and Constant Availability of Data

In general, only the most timely data resides on the higher-performance disk; DMF software automatically migrates less timely data to secondary storage. However, all of the data always appears to be online to users and applications using normal access methods, regardless of the data's actual location.

Although DMF software moves file **data**, it leaves file **metadata** in place so that users can access files without knowing the actual location of the data. *Metadata* consists of items such as index nodes (*inodes*) and directory structure. Migrated files appear as normal files to users and are always easily accessible via high-performance network connections.

Because migrated files remain cataloged in their original directories, users and applications never need to know where the data actually resides; they can access any migrated file using normal processes. In fact, when drilling into directories or listing their contents using standard POSIX-compliant commands, a user cannot determine the location of file data within the storage tier; determining the data's actual residence requires special commands or command options.

A file whose data blocks have been freed is considered from the **DMF software perspective** to be *offline* and its data blocks are therefore available for new active data, either new files or recalled files. However, from the **user perspective**, the file always appears to be online because the inodes and directories remain in the managed filesystem, allowing users to access the file by normal means.

The only difference users might notice when accessing a file whose data blocks have been freed is a delay in response time, because the data must be retrieved from secondary storage. From the user's perspective, all data always appears to be available online, regardless of its actual location.

Partial-State Files

Managed files can have multiple distinct file regions with different residency states. A *region* is a contiguous range of bytes that have the same residency state. A file that has more than one region is called a *partial-state* file. A file that is in a *static state* (that is, not currently being migrated or unmigrated) can have one region that is online in the managed filesystem for immediate access and another region that is offline and must be recalled in order to be accessed.

Partial-state files provide the following capabilities:

- *Accelerated access to first byte*, which allows you to access the beginning of an offline file before the entire file has been recalled.
- *Partial-state file online retention*, which allows you to keep a specific region of a file online while freeing the rest of it (for example, if you wanted to keep just the beginning of a file online). See "`ranges` Clause" on page 383.
- *Partial-state file recall*, which allows you to recall a specific region of a file without recalling the entire file. For more information, see the `dmput`(1) and `dmget`(1) man pages.

For additional details, see:

- "Modify Partial-State Capability with Care" on page 122
- "`dmdaemon` Object Parameters" on page 290
- Appendix G, "Considerations for Partial-State Files" on page 779

Safety and Scalability

DMF software transports large volumes of data on behalf of many users and has evolved to satisfy customer requirements for scalability and the safety of data:

- When you configure the DMF environment using best practices, DMF software creates at least two secondary-storage copies of the data in order to prevent file data loss in the event that a migrated copy is lost. See "Ensuring Data Integrity" on page 38.
- Because system interrupts and occasional storage device failures cannot be avoided, it is essential that the integrity of data be verifiable. Therefore, DMF software also provides tools necessary to validate your storage environment. See "Commands Overview" on page 56.
- DMF with the *Parallel Data-Mover Option* (referred to as *Parallel DMF*) lets you scale the DMF I/O capacity in cost-effective increments. A *data mover* is a node running processes that migrate and recall data to secondary storage. In the basic DMF product, the DMF server incorporates the functionality of an integrated data-mover node. Parallel DMF allows the DMF system to reside on a single server and minimizes the cost of a DMF implementation. For users with higher throughput requirements, this option allows multiple data movers to operate in parallel, increasing data throughput and enhancing resiliency. The parallel

data-mover node's dedicated function is to move data to and from secondary storage. See "Parallel DMF Overview" on page 33.

Site-Defined Migration Policies

As a DMF administrator, you determine how disk space capacity is handled by doing the following:

- Selecting the filesystems that DMF software will manage
- Specifying the amount of free space that will be maintained on each filesystem
- Ranking file-selection criteria, such as file size and file age

DMF software selects files for migration and frees data blocks of already migrated files based on site-defined criteria that are specified in a *migration policy*. For example, a migration policy does the following:

- Makes the specified number of copies of migrated data. DMF software places those copies on separate secondary-storage targets. SGI strongly recommends that you create at least two secondary-storage copies in order to prevent file data loss in the event that one copy is damaged.
- Migrates the data at the times specified or when the specified free-space minimum threshold is exceeded.
- Optionally keeps a small amount of data in the managed filesystem for each file, even after migration (for use by file managers, in order to avoid unnecessary recall of a file due to directory browsing).
- Maintains a specified percentage of the managed filesystem free for new data (either new files or recalled files). When the filesystem reaches this threshold, DMF software will free the already-migrated data blocks until the specified percentage of the filesystem is free, normally selecting files by size and last-access time.

A Variety of Migration Targets

DMF software can migrate data to the following media:

- Cloud storage
- Fibre Channel tapes and tape libraries that are supported by the OpenVault or TMF mounting services

- SCSI low-voltage differential (LVD) tapes and tape libraries

  Note: If you have a high-voltage differential (HVD) tape or tape library that you want to use for the DMF environment, you must contact SGI Professional Services for assistance in obtaining the appropriate HVD-LVD converter.

  The LVD requirement is only for tapes and tape libraries. It does not apply to HVD disk.

- Disk
- COPAN RAID sets:
  - COPAN massive array of idle disks (MAID)
  - SGI 400 virtual tape library (VTL)
- JBFS configurations
- Another server (via NFS or FTP)

You can also use disk or COPAN RAID sets as a cache in conjunction with another migration target to provide multiple levels of migration; see "Multiple Storage Tiers" on page 19.

Support for Fileserving Applications

DMF software supports a range of storage-management applications. In some environments, DMF software is used strictly to manage highly stressed online disk resources. In other environments, it is also used as an organizational tool for safely managing large amounts of data. In all environments, DMF software scales to the storage application and to the characteristics of the available storage devices.

DMF software interoperates with the following:

- Standard data export services such as NFS and FTP
- XFS® filesystems
- CXFS™ clustered filesystems
- Microsoft® SMB (also known as the CIFS) as used by Samba when fileserving to Windows® systems

By combining these services with DMF software, you can configure an SGI system as a high-performance fileserver.

DMF Manager Web Interface

DMF software provides a set of graphical and command-line tools to help you configure, monitor, and manage the DMF system. *DMF Manager* is a web-based tool you can use to do the following:

- Configure the DMF environment
- Install DMF licenses
- Display status of the DMF environment
- Display reports about internal DMF processing queues, allowing you to cancel and reprioritize specific requests
- Start and stop DMF processes
- Deal with day-to-day DMF operational issues
- Display performance metrics, including filesystem throughput and volume usage
- Create custom statistics reports
- Accommodate tape volumes that are physically not in the tape library
- Show SGI Linear Tape File System (LTFS) information, configure LTFS, and mount/unmount LTFS tapes
- Restore filesystems and filesystem components

For details, see:

- "DMF Manager Requirements" on page 49
- Chapter 6, "DMF Manager" on page 191
- Chapter 17, "DMF Queue Viewing" on page 605

Also see:

- "Using Out-of-Library Tapes" on page 184
- Chapter 18, "SGI LTFS" on page 631

DMF Manager is useful for all DMF customers from enterprise to high-performance computing and is available via the Firefox® and Internet Explorer® web browsers.

At a glance, you can see if the DMF environment is operating properly. An icon in the upper-right corner indicates if the DMF environment is up (green) or down (upside down and red). If something requires, DMF Manager makes actions available to identify and resolve problems. The tool volunteers information and provides context-sensitive online help. DMF Manager also displays performance statistics, allowing you to monitor DMF activity, filesystems, and hardware.

Figure 1-3 is an example of the **Overview** panel. It shows status of the DMF environment, including the following:

- The DMF environment is up (green icon)
- There are some warnings that may require action (yellow icon)
- The `/dmi_fs2` filesystem is related to the `volume1` and `volume2` volume groups (VGs)

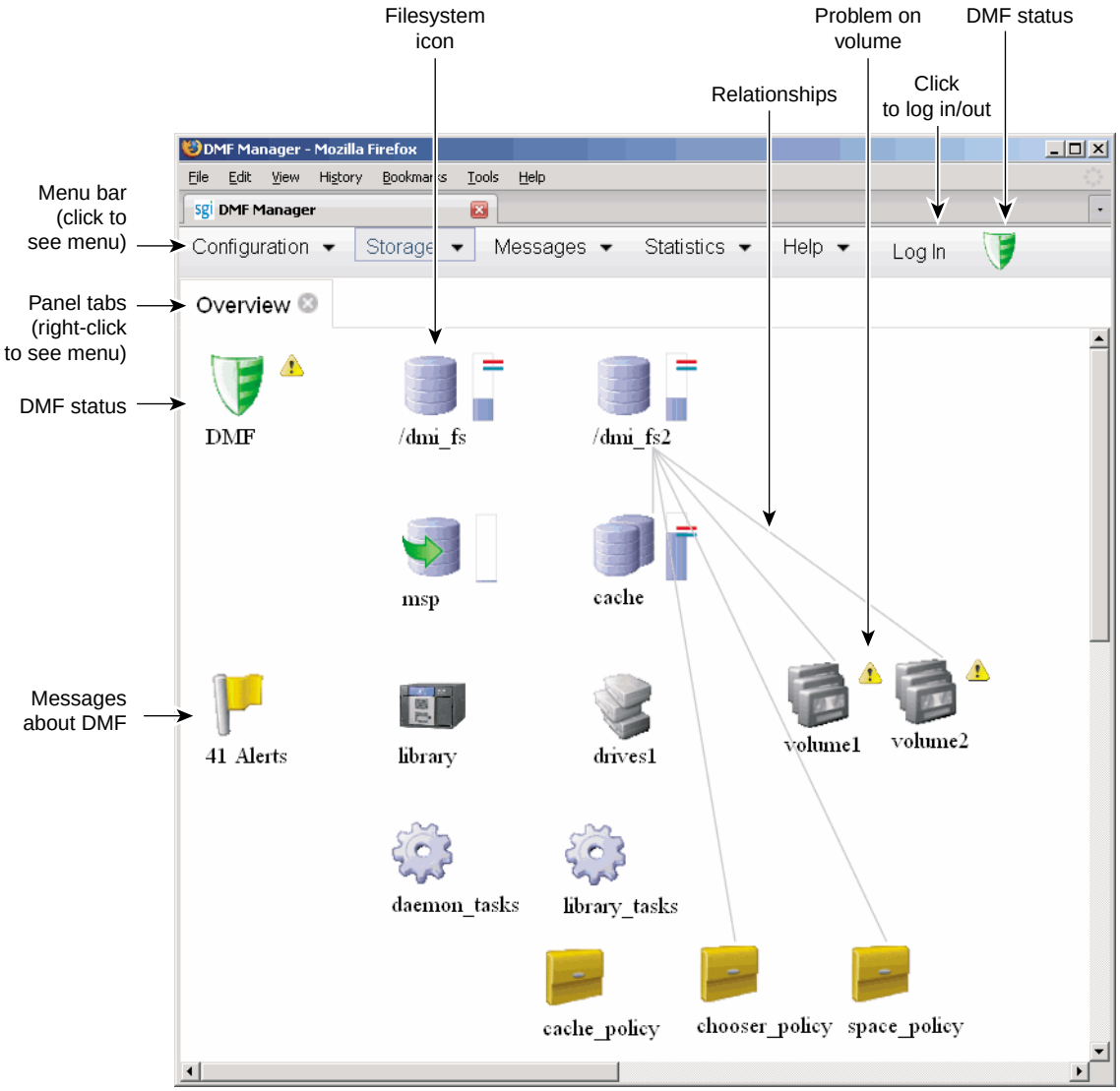

Figure 1-3 DMF Manager

DMF Control from Client Platforms

This section discusses the following:

- "Overview of the Windows Client" on page 12
- "Overview of the IRIX, Mac OS X, Linux, and Solaris Clients" on page 12

Overview of the Windows Client

The DMF client for Windows systems lets users and administrators control DMF via file shares configured on the Samba server. The Samba server must have SGI enhanced Samba installed, and may be either the DMF server or a CXFS client-only node.

Using Windows Explorer, you can do the following for files on which you have the appropriate permission, depending upon site-specific configuration:

- View DMF file properties
- Execute the following DMF user operations:
  - Recall files, similar to the functionality of the `dmget`(1) command
  - Migrate files, similar to the functionality of the `dmput`(1) command
  - Set site tags, similar to the functionality of the `dmtag`(1) command
- Set a project ID, similar to functionality of the `dmprojid`(8) command

See *DMF 6 Client Guide for Windows Systems.*

Overview of the IRIX, Mac OS X, Linux, and Solaris Clients

Several DMF user commands are available natively on DMF clients running any of the following operating systems (see the DMF release notes for the specific versions that are supported):

- SGI IRIX®
- Apple® Mac OS X®
- Red Hat® Enterprise Linux® (RHEL)
- SUSE® Linux® Enterprise Server (SLES)
- Sun™ Solaris™

For more details, see "User Commands" on page 57.

High Availability

You can run DMF software in a high-availability (HA) cluster.

Caution: This will require some configuration requirements and administrative procedures (such as starting/stopping DMF services) that differ from the information in this DMF guide. For more information about DMF and HA, see *High Availability Guide for SGI InfiniteStorage*.

SOAP Web Service

DMF software provides access to a subset of the DMF client functions via the DMF Simple Object Access Protocol (SOAP) web service. For more information, see Chapter 16, "DMF SOAP Server" on page 597.

Direct Archiving

You can use the *direct archiving* feature to manually copy file data between a POSIX filesystem (such as a Lustre™ filesystem) directly to DMF secondary storage by configuring the POSIX filesystem for archive use in the DMF configuration file and using the `dmarchive`(1) command. The POSIX filesystem cannot be DMAPI-enabled (that is, it cannot be mounted with the `dmi` mount option) and is known as an *archive filesystem*. When using this feature, DMF software copies the file data to DMF secondary storage while placing the metadata in a filesystem that is managed by DMF. See "Use `dmarchive` to Copy Archive File Data to Secondary Storage" on page 119.

Mounting Services

When you purchase DMF software, you also receive the following mounting services:

- OpenVault storage library management facility, applicable to SLES or RHEL. See *OpenVault Administrator Guide for SGI InfiniteStorage*.
- Tape Management Facility (TMF), applicable to SLES only. See *TMF 6 Administrator Guide for SGI InfiniteStorage*.

Out-of-Library Tapes

When OpenVault is the mounting service, DMF software will try to retrieve data from an in-library volume before requesting that an out-of-library tape be imported. See "Using Out-of-Library Tapes" on page 184.

How DMF Software Works

This section discusses the following:

- "File States" on page 14
- "DMF Methods" on page 15
- "Multiple Storage Tiers" on page 19
- "Migration Process" on page 27
- "Recall of File Data" on page 27
- "Fast-Mount Cache Configuration Overview" on page 30
- "DMF Server Functions" on page 32
- "Parallel DMF Overview" on page 33
- "DMF Databases" on page 37
- "Ensuring Data Integrity" on page 38
- "DMF Architecture" on page 38
- "Migrate Groups" on page 42
- "DMF Capacity" on page 42

File States

DMF software uses the following terminology with regard to the state of a file in a managed filesystem:

- *Regular file* (REG) is a file residing only on the high-performance disk in the managed filesystem.

- *Migrating file* (MIG) is a file whose copies on secondary storage are in progress.
- *Migrated file* is a file that has one or more complete copies on secondary storage and no pending or incomplete offline copies. A migrated file is one of the following from the perspective of DMF software:
  - *Dual-state file* (DUL) is a file whose data resides both on the high-performance disk and on secondary storage
  - *Offline file* (OFL) is a file whose data is no longer on the high-performance disk (the data is offline from the DMF perspective, but from the user perspective the data always appears to be available online)
  - *Unmigrating file* (UNM) is a previously offline file in the process of being recalled to the high-performance disk
  - *Partial-state file* (PAR) is a file with some combination of dual-state, offline, and / or unmigrating regions

When a file is first migrated, DMF software copies the data to secondary storage but may not immediately free the data in the managed filesystem on the high-performance disk. During this period, the file is considered to be *dual-state* because it resides in both locations. Like a regular file, a migrated file has an inode. An offline file or a partial-state file requires the intervention of the DMF daemon to access its offline data; a dual-state file is accessed directly from the original that still exists in the managed filesystem.

The operating system informs the DMF daemon when a migrated file is modified. If anything is written to a migrated file, the offline copy is no longer valid, and the file becomes a regular file until it is migrated again.

If you are using DMF direct archiving to copy files from a filesystem that is not managed, *archiving files* are files where the original resides on an archive filesystem (one not managed by DMF software, such as Lustre) and whose offline copies are in progress. When the process completes, the files are offline files.

DMF Methods

The migration process is managed by a daemon-like component called a *library server (LS)* or *media-specific process (MSP)*:

- *LS* (dmatls) transfers data to and from the following types of volumes:
  - Magnetic tape in a tape library (also known as a *robotic library* or *silo*)

  - RAID sets in a COPAN MAID system[1]
  - Virtual tapes in an SGI 400 VTL system
  - JBFS configurations

- *Cloud MSP* (`dmcloudmsp`) transfers data to and from a cloud storage system accessible via a network (local or Internet).
- *FTP MSP* (`dmftpmsp`) uses the File Transfer Protocol to transfer data to and from disks of another system on the network.
- *Disk MSP* (`dmdskmsp`) uses a filesystem mounted on the DMF server itself as the location on which to store/recall file data. See "Use an Appropriate Filesystem for a Disk MSP" on page 121.
- *Disk cache manager (DCM) MSP* is the disk MSP configured for *n*-tier capability by using a dedicated filesystem as a cache. DMF software can manage the disk MSP's storage filesystem and further migrate it to secondary storage, thereby using a slower and less-expensive dedicated filesystem as a cache to improve the performance when recalling files. DCM MSP configuration generally first migrates data to cache on (for example) serial ATA (SATA) disk and then at a later time migrates the data from the SATA disk to secondary-storage on physical tape. The filesystem used by the DCM MSP must be a local XFS or CXFS filesystem.
- *Fast-mount cache configuration* is a special configuration of an LS volume group that simultaneously migrates data to a copy on the cache target (such as COPAN MAID or JBFS configurations) with rapid mount and positioning characteristics and to secondary-storage copies on the other targets (such as physical tape). This configuration provides similar functionality to a DCM MSP but does not downwardly migrate data from the cache tier; in this configuration, an entire volume on the cache can be freed immediately when the fullness threshold is reached. See "Fast-Mount Cache Configuration Overview" on page 30.

A site can use any combination of DMF methods.

Figure 1-4 and Figure 1-5 summarize these concepts and "Multiple Storage Tiers" on page 19 provides more details and illustrations.

1 For historical reasons, these volumes are sometimes referred to as *tapes* in command output and documentation.

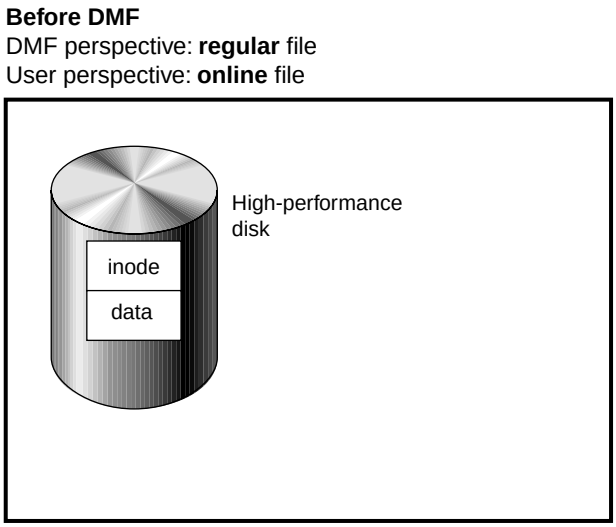

Figure 1-4 DMF Methods: Before Migrating

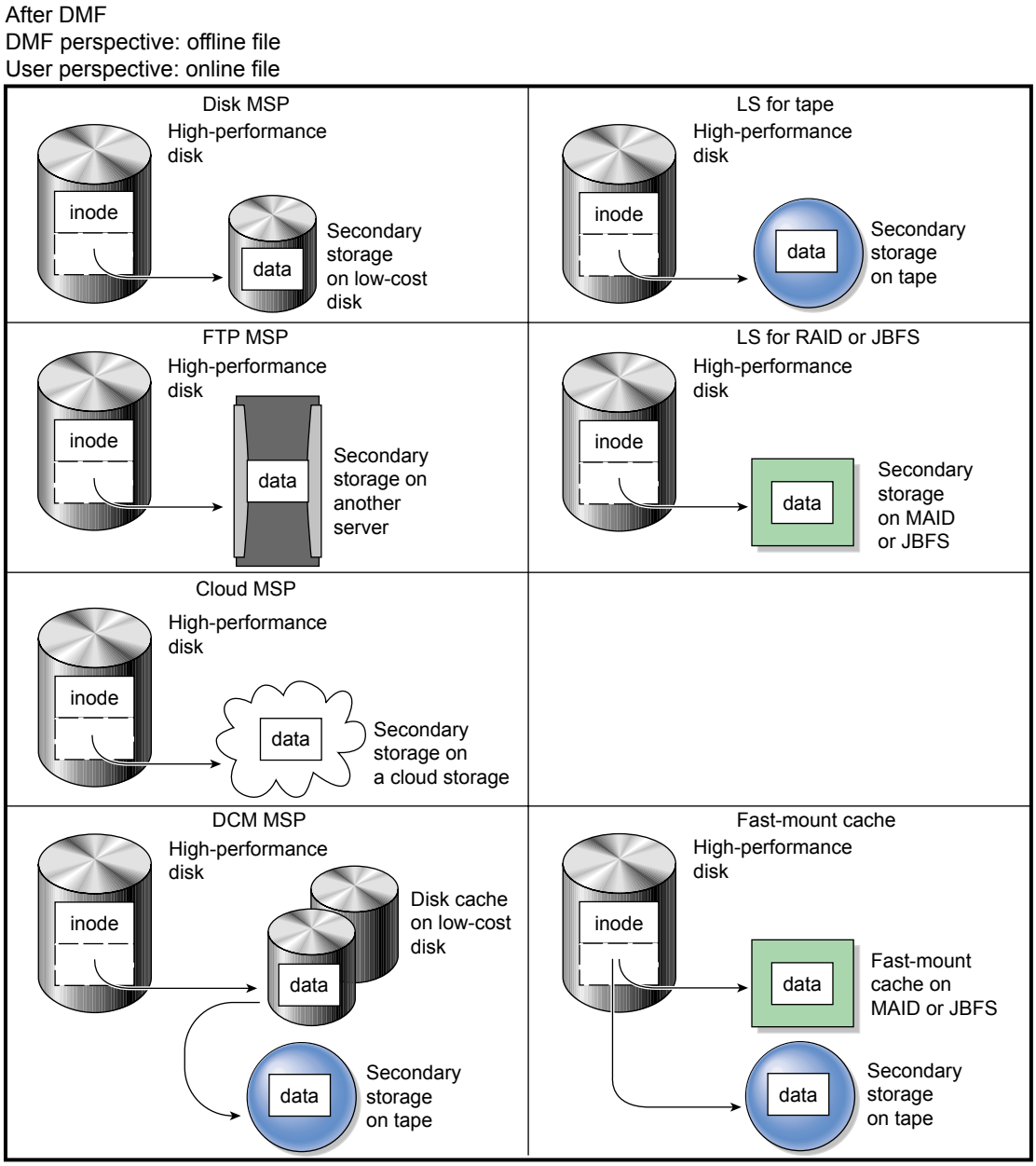

Figure 1-5 DMF Methods: After Migrating Data and Freeing Space

Multiple Storage Tiers

The various DMF methods provide multiple storage tiers:

- "Two Tiers" on page 19
- "Three Tiers using DCM MSP" on page 21
- "Three Tiers using Fast-Mount Cache" on page 24

The figures in the following subsections show the use of multiple tiers and the concepts of DMF data migration (in which file data is copied from the managed filesystem to the secondary storage, but the inode remains in place in the managed filesystem) and data recall.

Note: For simplicity, the figures in this chapter do not address a second copy of secondary storage. Data will be recalled from a second copy only if necessary.

Two Tiers

LS and *non-cache MSPs* (cloud MSP, disk MSP, or FTP MSP) provide two tiers of storage media:

- Tier-1: Managed filesystem on high-performance disk
- Tier-2: Secondary storage on cloud storage, disk (including COPAN MAID, COPAN VTL, and JBFS configurations), FTP server, or tape

Figure 1-6 and Figure 1-7 show an example of the process using two tiers.

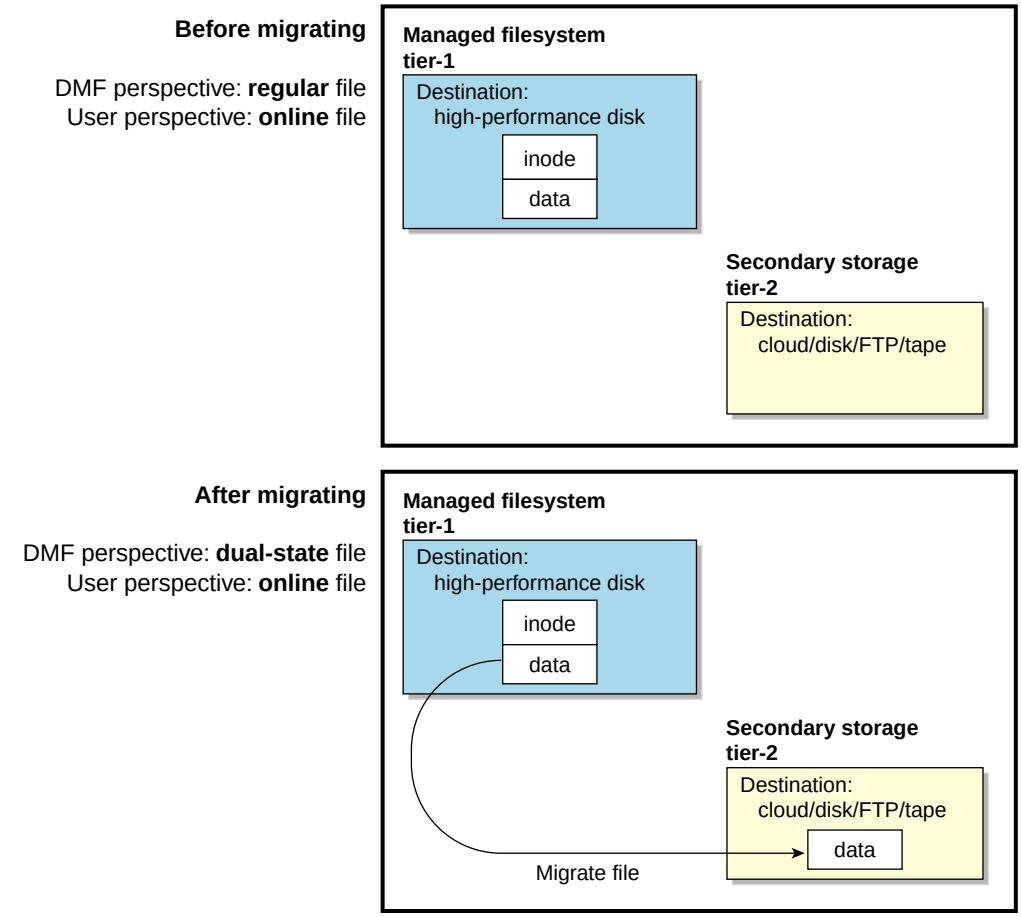

Figure 1-6 Two Tiers: Migrating File Data

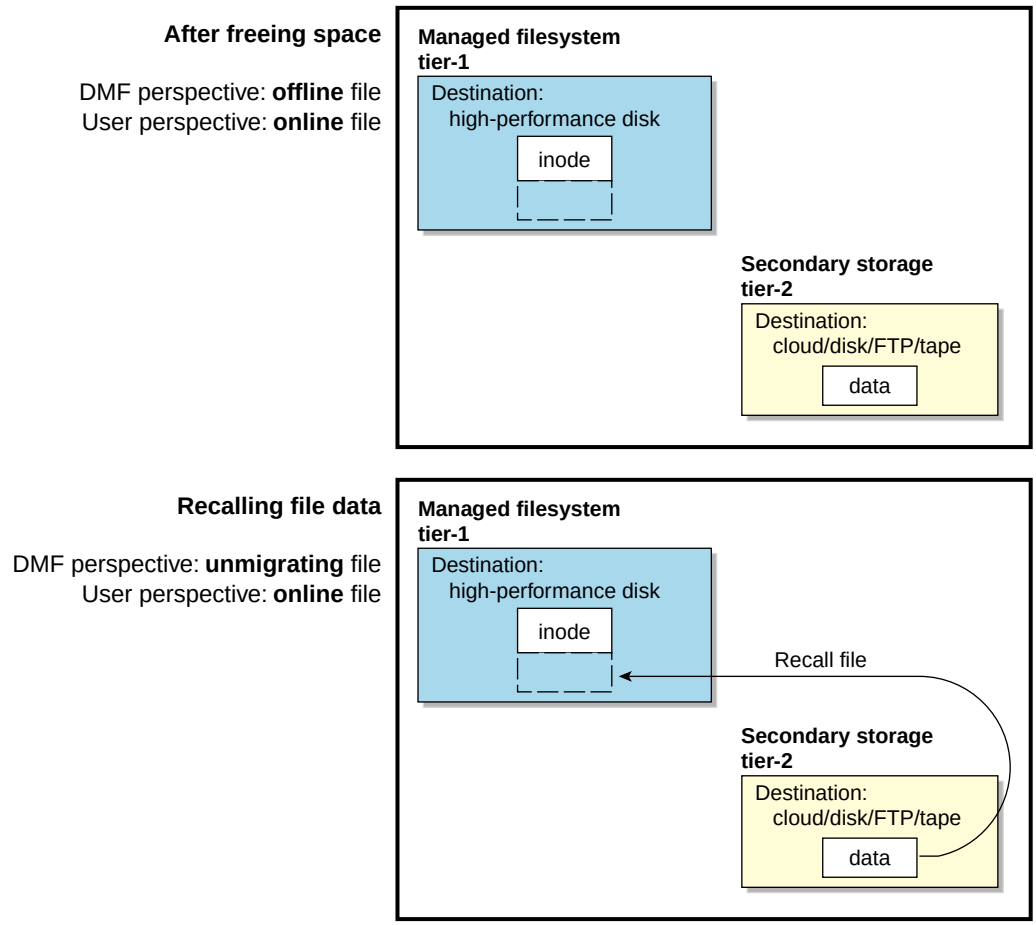

Figure 1-7 Two Tiers: Freeing and Recalling File Data

Three Tiers using DCM MSP

Adding a DCM MSP provides three tiers of storage media:

- Tier-1: Managed filesystem on high-performance disk
- Tier-2: Cache on high-capacity, low-cost disk that will downwardly migrate and free data on a file basis
- Tier-3: Secondary storage on cloud storage, FTP server, or tape

Figure 1-8 and Figure 1-9 show an example of the process using three tiers of storage with a DCM MSP, where data moves first to a cache on lower-performance but less-expensive disk, then to inexpensive storage. The file will be recalled from disk cache as long as it resides there because it is faster than recalling from the third tier.

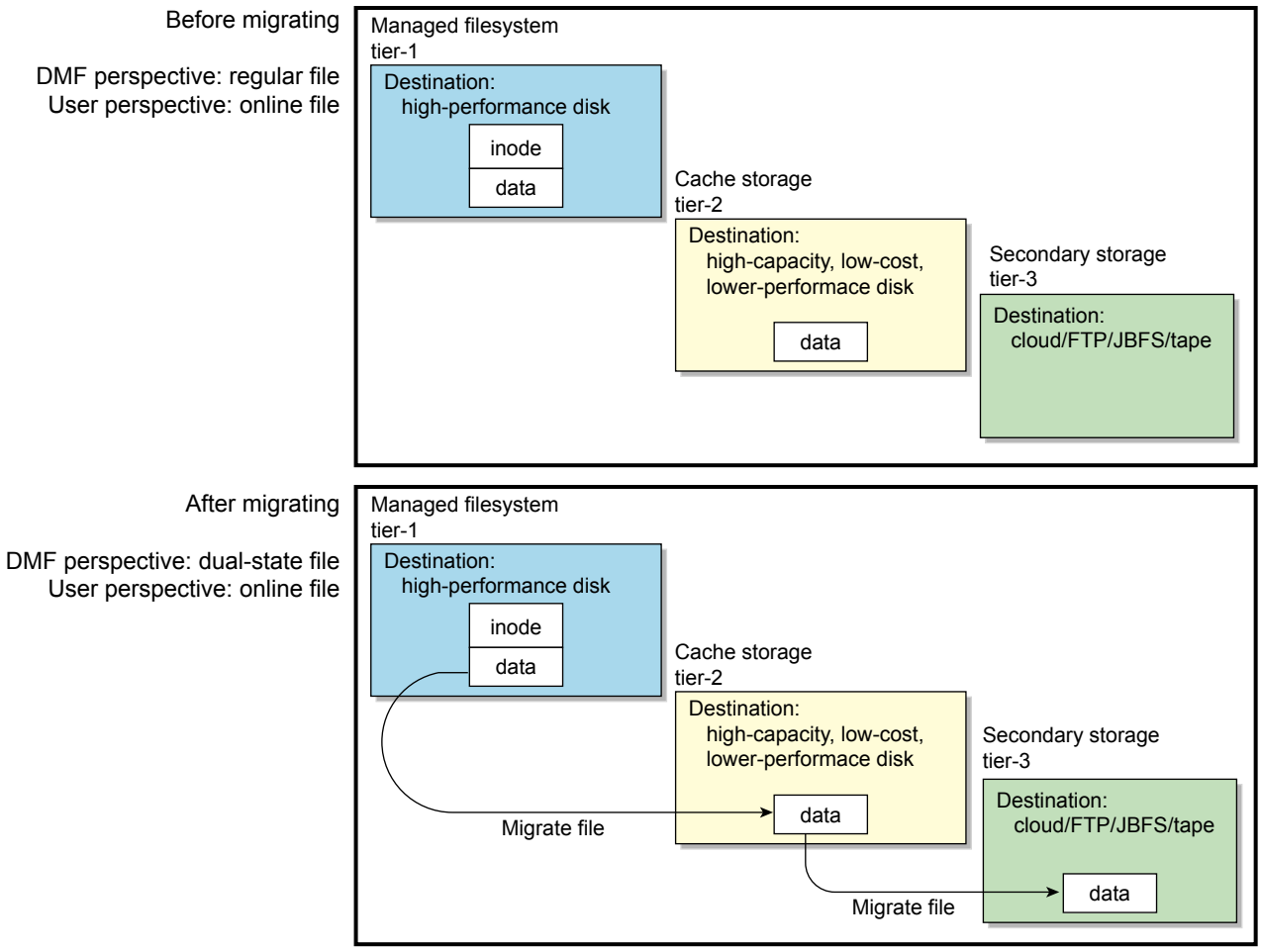

Figure 1-8 DCM MSP: Migrating File Data

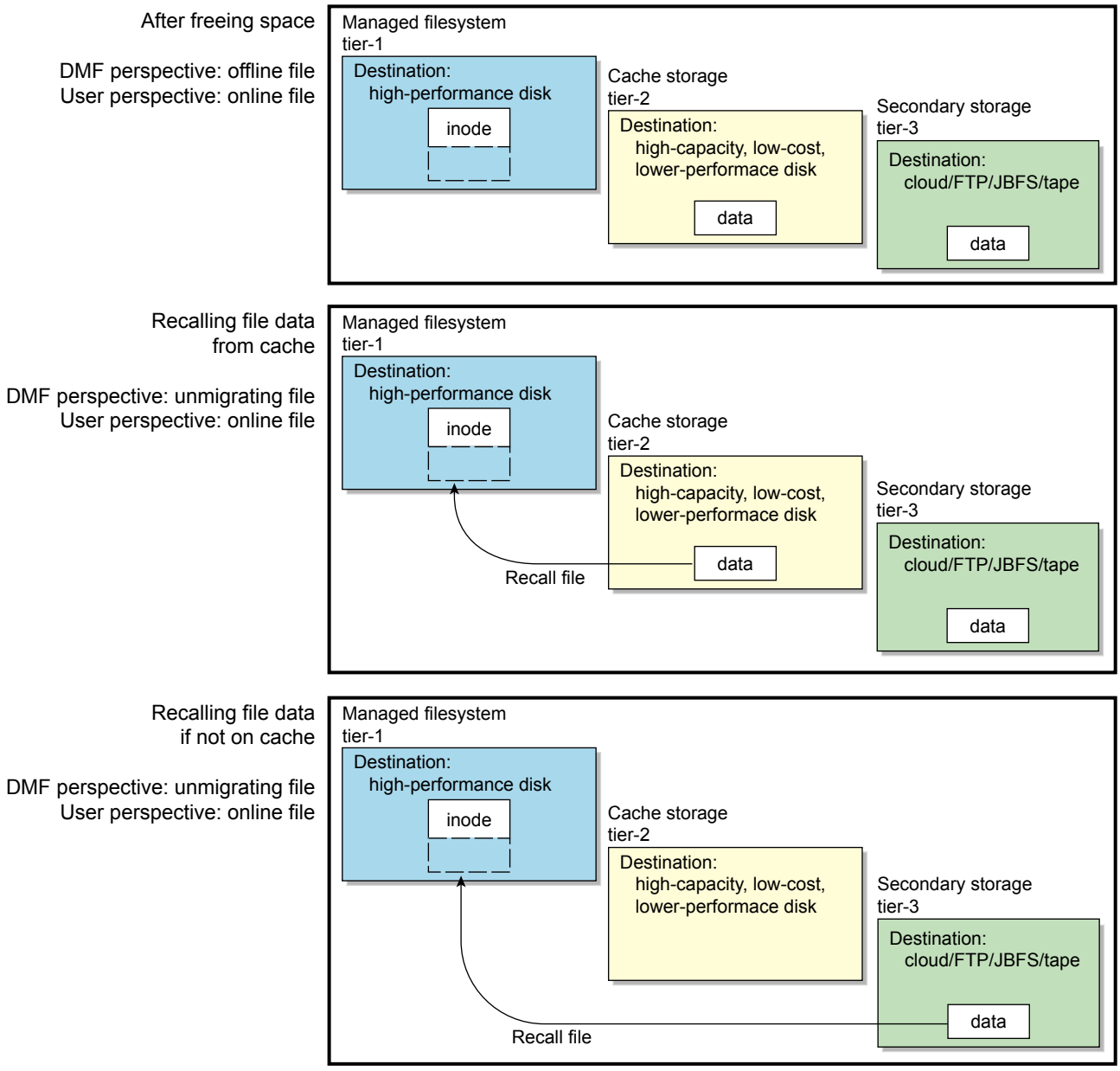

Figure 1-9 DCM MSP: Freeing and Recalling File Data

Three Tiers using Fast-Mount Cache

Adding a fast-mount cache provides three tiers of storage media:

- Tier-1: Managed filesystem on high-performance disk
- Tier-2: Fast-mount cache (such as COPAN MAID or JBFS configurations) that will be freed on a volume basis (no downward migration)
- Tier-3: Secondary storage on cloud storage, FTP server, JBFS configurations, and tape

Figure 1-10 and Figure 1-11 show an example of the process using three tiers of storage, where a copy of the data is simultaneously placed in tier-2 fast-mount cache (such as COPAN MAID or JBFS configurations) and in tier-3 secondary storage (such as tape). The file will be recalled from the cache as long as it resides there because it is faster than recalling from tier-3 storage.

Note: Unlike the DCM MSP, this method does not migrate data from the cache to tier-3; therefore, volumes on the cache can be freed immediately when the fullness threshold is reached.

For more information, see "Fast-Mount Cache Configuration Overview" on page 30.

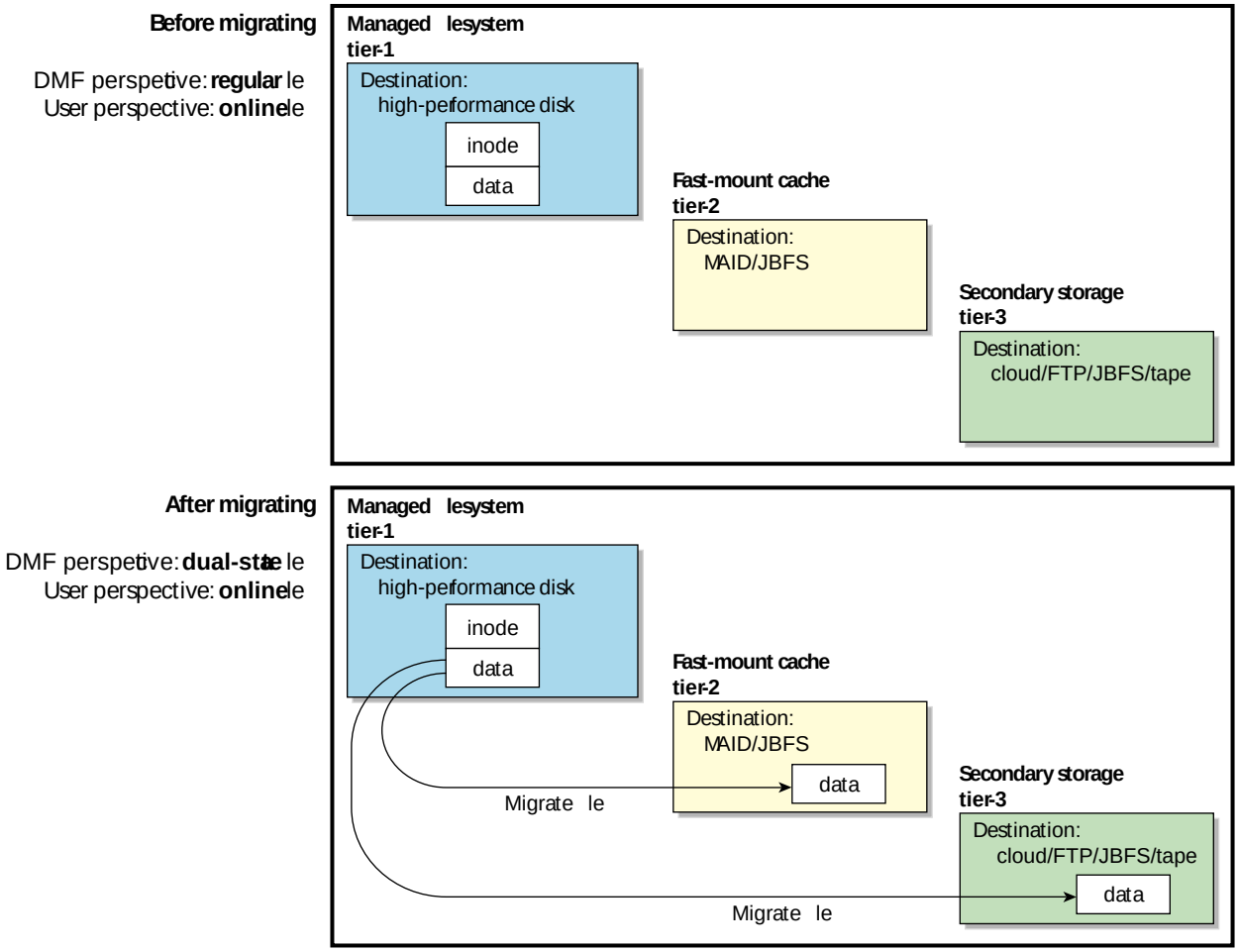

Figure 1-10 Fast-Mount Cache: Migrating File Data

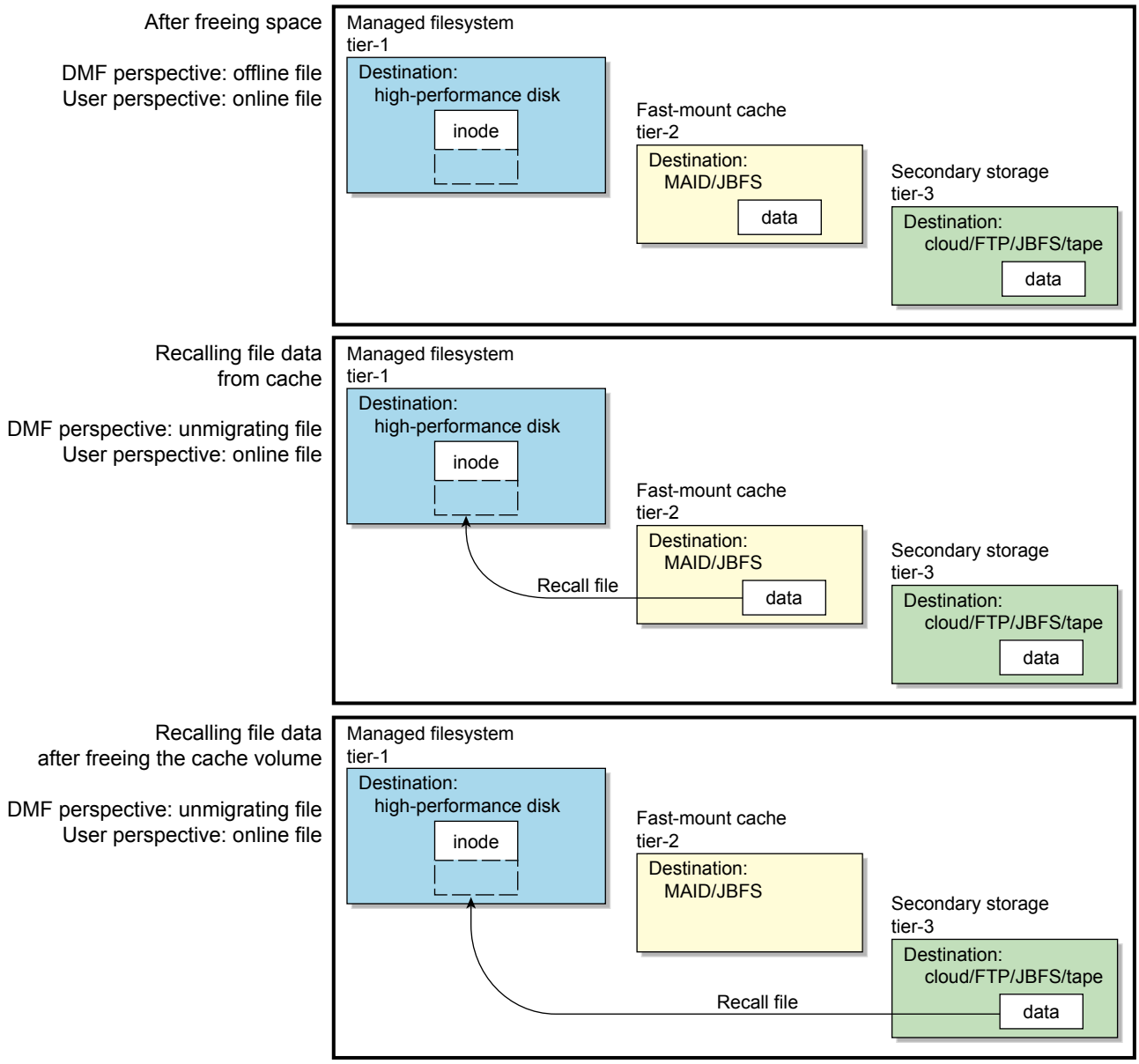

Figure 1-11 Fast-Mount Cache: Freeing and Recalling File Data

Migration Process

You choose both the percentage of the filesystem to migrate and the amount of free space. You as the administrator can manually trigger file migration or file owners can issue manual migration requests.

A file is migrated when the automated space-management controller `dmfsfree`(8) selects the file or when an owner requests that the file be migrated by using the `dmput`(1) command.

When the daemon receives a request to migrate a file, it does the following:

1. Adjusts the state of the file.
2. Ensures that the necessary MSPs/VGs are active.
3. Sends a request to the MSPs/VGs, who in turn copy data to the secondary storage media.

When the MSPs/VGs have completed the offline copies, the daemon marks the file as migrated in its database and changes the file to dual-state. If the user specifies the `dmput -r` option, or if `dmfsfree` requests that the file's space be released, the daemon releases the data blocks and changes the file state to offline. For more information, see the `dmput`(1) man page.

Note: DMF software does not migrate pipes, directories, or UNIX® or Linux special files.

Recall of File Data

This section discusses the following:

- "Recall from the Appropriate Location" on page 27
- "Order of Recall" on page 28
- "Freeing Data Blocks to Make Space for Recalls" on page 30

Recall from the Appropriate Location

Data is provided to the user from the appropriate location:

- If a user accesses a dual-state file, the data comes directly from the high-performance disk as normal, providing the fastest access.
- After the data blocks on the managed filesystem are freed, DMF software automatically recalls the file's data from the secondary storage when the user accesses the file, placing the data back on the managed filesystem; at this point, the file once again becomes a dual-state file. (If the user then changes the file, it returns to being a regular file.)

Order of Recall

When a migrated file must be recalled, a request is made to the DMF daemon. The daemon selects an MSP/VG from its internal list and sends that MSP/VG a request to recall a copy of the file. If more than one MSP/VG has a copy, the first one in the list is used. The list is created from the configuration file.

For illustration purposes, suppose that the DMF configuration file contains the following definitions for an environment using a single library server with two drive groups that each have two volume groups to specify the location of file copies:

- The `dmdaemon` object contains the following parameter to identify the library server:

  ```
  LS_NAMES          myls
  ```

- The `libraryserver` object defined for `myls` contains the following parameter to identify the drive groups and their order of selection:

  ```
  DRIVE_GROUPS     fruits veggies
  ```

- The `drivegroup` objects defined for `fruits` and `vegetables` identify their respective volume groups and their order of selection:

  – For `fruits`:

  ```
  VOLUME_GROUPS     oranges apples
  ```

  – For `veggies`:

  ```
  VOLUME_GROUPS     carrots peas
  ```

The order in which volumes are chosen for recall is decided by the order in which drive groups and then volume groups are listed in their respective definitions. Given the above, `oranges` will be tried first and `peas` will be tried last. If you stopped

DMF, reordered the list in DRIVE_GROUPS, and restarted DMF, then carrots would be tried first and apples be tried last, as shown in Figure 1-12.

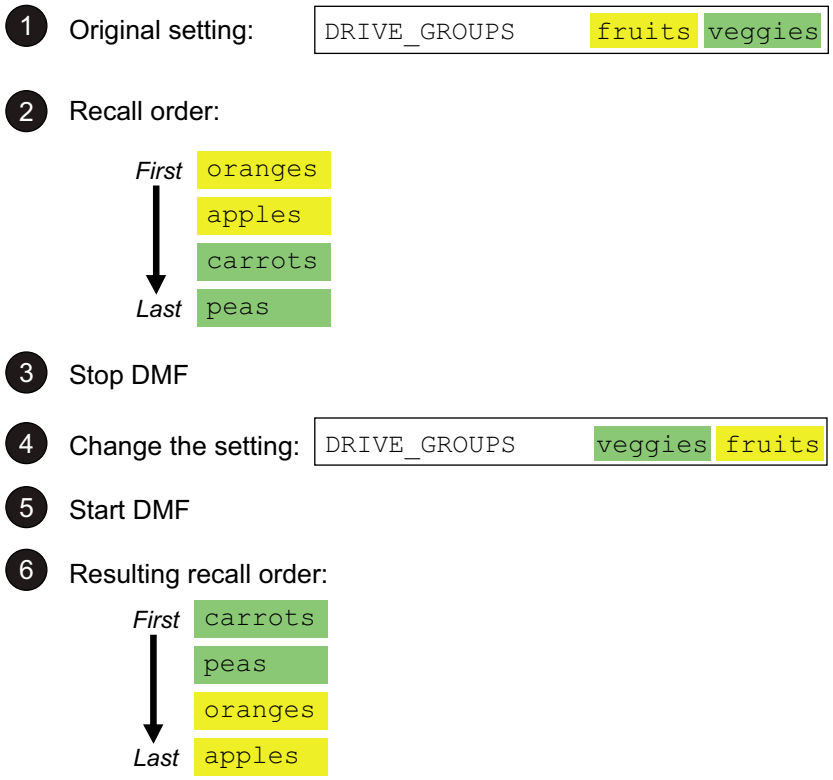

Figure 1-12 Recall Order

Note: You must not change these parameters while DMF is running.

For more details, see Chapter 7, "DMF Configuration File" on page 271, specifically:

- "dmdaemon Object Parameters" on page 290
- "libraryserver Object Summary" on page 393
- "drivegroup Object Parameters" on page 397

Freeing Data Blocks to Make Space for Recalls

If you recall more files than the managed filesystem can currently contain, DMF software migrates other files and will free the data blocks of already-migrated files (according to site-specific policies) until the filesystem is once again well below the free-space minimum threshold.

Note: A file's data blocks on the managed filesystem can only be freed **after** the data has been copied to secondary storage.

Fast-Mount Cache Configuration Overview

This section discusses the following:

- "Cache and Secondary-Storage Targets" on page 30
- "How Fast-Mount Cache Differs from a DCM MSP" on page 31
- "Fast-Mount Cache Implementation" on page 32
- "Appropriate Use of Fast-Mount Cache" on page 32

Cache and Secondary-Storage Targets

You can use a cache migration target with rapid mount and positioning characteristics in conjunction with other secondary-storage targets in a fast-mount cache configuration. For example, consider the following:

- COPAN MAID and JBFS configurations are faster than physical tapes, but their storage size is finite
- A physical tape library has an effectively unlimited storage capacity because you can eject full tapes and replace them with empty tapes, but recalling data from tape is slower than recalling data from COPAN MAID or JBFS configurations

The combination of these two targets in a fast-mount cache configuration results in faster recall performance for recently created offline files while also providing secure long-term storage.

How Fast-Mount Cache Differs from a DCM MSP

A fast-mount cache is similar to a DCM MSP in that both provide fast recall of migrated files in the cache tier (tier-2). However, they have following important differences:

- DCM MSP:
  - Can be configured to downwardly migrate data from tier-2 to tier-3 as the data ages
  - Only requires that one initial copy be made, although two copies are recommended to prevent data loss (the copy in cache can be downwardly migrated to secondary storage on tier-3)
  - Deletes data from tier-2 on an individual file basis
  - Data on tier-2 may not be immediately recoverable when space is needed if the data does not already have a copy in tier-3 (causing a delay if space is needed quickly)
- Fast-mount cache:
  - Does not downwardly migrate data from tier-2 to tier-3
  - Always requires that at least two initial copies be made (a copy to the cache and a copy to the secondary storage on tier-3)
  - Deletes data from tier-2 on a volume basis (that is, all files in the volume are deleted at the same time)
  - Tier-2 can be freed immediately when the free-space threshold is reached, without further operational effort

---

Note: SGI strongly recommends that you migrate at least two copies to secondary-storage targets in order to prevent file data loss in the event that a migrated copy is damaged. When using a fast-mount cache, SGI therefore recommends that you migrate at least three copies (one copy to the cache on tier-2 and two copies to secondary-storage targets at the tier-3 level).

Fast-Mount Cache Implementation

To implement a fast-mount cache, you must configure the DMF environment to make all secondary-storage copies of the data (tier-3 storage on other MSPs/VGs) at the same time as the cache copy (tier-2 storage on the MGs/VGs in the fast-mount cache).

You must also configure a task to empty the fast-mount cache when it reaches the configurable free-space threshold. DMF software immediately empties the oldest full volumes, defined as those with the oldest write dates. Because at least one copy of the data exists elsewhere (most likely on a physical tape), there is no need to wait for the data in the disk cache to migrate to a lower tier (unlike a DCM MSP). Therefore, the freeing of space on the fast-mount cache is very fast because it requires no movement of data.

Figure 1-10 on page 25 and Figure 1-11 on page 26 summarize the concepts of migrating and recalling file data in a fast-mount cache configuration using COPAN MAID as an example.

Also see "Use Fast-Mount Cache Appropriately" on page 112.

Appropriate Use of Fast-Mount Cache

The fast-mount cache configuration is most appropriate for sites that have a high turnover of often-accessed data, where the most recently migrated files are also the most likely to be recalled.

All files on a volume being freed are deleted without regard to their size or last access time. That might mean that a file that is still being actively recalled on a fairly regular basis must be recalled from a VG with slower mount and position characteristics. You can minimize this issue by setting optional configuration parameters so that recently accessed files are copied to another volume within the fast-mount cache before any volumes are freed, using a separate scratch directory, but there may be an associated performance impact.

DMF Server Functions

The DMF server always provides the following services:

- DMF administration (see "Administration Tasks" on page 51)
- Backups

- All I/O for data transfer to and from disks that is associated with cloud, FTP, disk, or DCM MSPs (see "How DMF Software Works" on page 14)
- By default, a portion of I/O for data transfer to and from secondary storage (using its integrated data-mover functionality)

Parallel DMF Overview

The individual processes that migrate and recall data are known as *data-mover processes*. Nodes that run data-mover processes are *data movers*; this may include the DMF server node if it is configured to use the *integrated data-mover functionality* and, if you have purchased the *Parallel Data-Mover Option*, the *parallel data-mover nodes*. The DMF server and the parallel data-mover nodes can each run multiple data-mover processes.

As shown in Figure 1-13, the *basic DMF* product (that is, without the Parallel Data-Mover Option) runs data-mover processes on the DMF server. This allows the DMF control system to reside on a single server and minimizes the cost of a DMF implementation. Additional nodes can be installed with DMF client software (see "DMF Control from Client Platforms" on page 12).

Figure 1-14 shows the DMF product in a CXFS clustered filesystem environment.

Note: All nodes connect to a network. For simplicity, the network and DMF clients are not shown in the following figures.

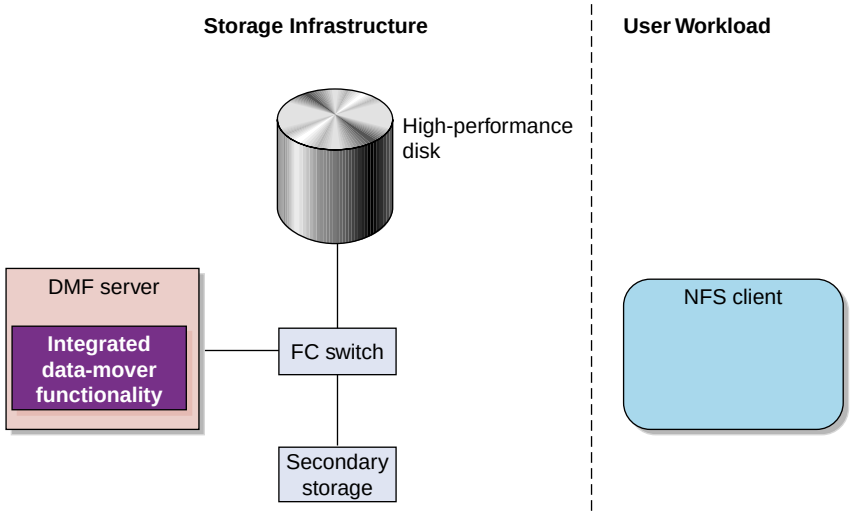

Figure 1-13 Basic DMF Product in an NFS Environment

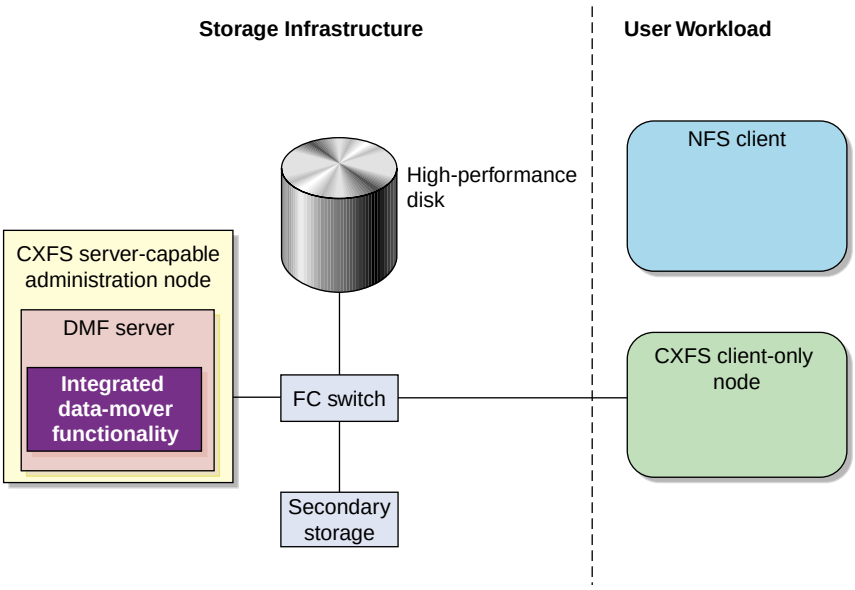

Figure 1-14 Basic DMF Product in a CXFS Environment

For users with higher throughput requirements, Parallel DMF allows additional data movers to operate in parallel with the integrated data-mover functionality on the DMF server, increasing data throughput and enhancing resiliency.

The parallel data-mover node's dedicated function is to move data from the managed filesystem to *volume-based media* (COPAN MAID, COPAN VTL, JBFS configurations, or tape) back into the managed filesystem, using an LS. Offloading the majority of I/O from the integrated data-mover functionality on the DMF server improves I/O throughput performance.

Because multiple parallel data-mover nodes can be used to move data, DMF software can scale its I/O throughput capabilities. When one parallel data-mover node hits its peak throughput capabilities, you can add more parallel data-mover nodes to the configuration as needed to improve I/O performance. Each parallel data-mover node can improve overall DMF performance by up to its maximum performance. For example, if you have parallel data-mover nodes that each provide up to a 2-GB/s increase, then having a configuration with three of these parallel data-mover nodes would provide a net increase of up to 6 GB/s. Additional drives and filesystem bandwidth may be required to realize the benefit from additional parallel data-mover nodes.

The basic DMF product can run in an environment with or without CXFS. If DMF software is managing a CXFS filesystem, DMF software will ensure that the filesystem's CXFS metadata server is on the same machine as the DMF server and will use metadata server relocation if necessary to achieve that configuration (see "Configure DMF Appropriately with CXFS™" on page 104). Parallel DMF must always run in a CXFS environment. The parallel data-mover nodes are SGI x86_64 machines that are installed with the **SGI DMF Parallel Data Mover** software package, which includes the required underlying CXFS software.

Note: From the CXFS cluster point of view, a DMF parallel data-mover node is a CXFS client-only node and therefore counts towards the total number of CXFS cluster nodes. However, the parallel data-mover nodes must be dedicated to DMF data-mover activities; they cannot perform any other functions that would be normal for CXFS client-only nodes.

The parallel data-mover node has specific hardware requirements and must access volume-based media on a port that is not used by CXFS. See "SAN Switch Zoning or Separate SAN Fabric Requirement" on page 49.

If you choose the Parallel DMF, you must use OpenVault for those drive groups (DGs) that contain drives on parallel data-mover nodes.

Figure 1-15 shows the concept of the DMF product using parallel data-mover nodes in a CXFS cluster with only one server-capable administration node. The parallel data-mover nodes only write data to secondary storage on volume-based media in an LS.

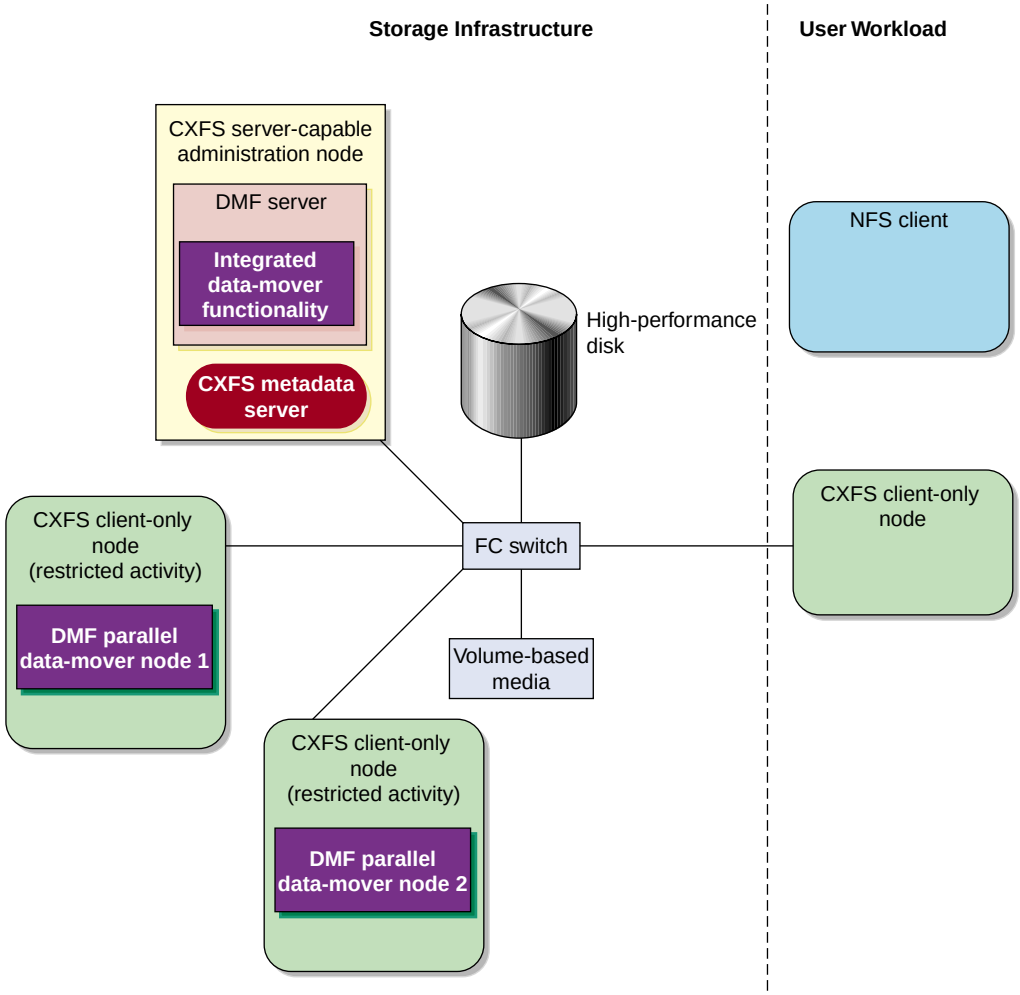

Figure 1-15 Parallel DMF in a CXFS Environment

In a configuration with Parallel DMF, the DMF server still provides the services listed in "DMF Server Functions" on page 32.

For more information, see Chapter 8, "Parallel DMF Configuration" on page 485.

DMF Databases

The DMF daemon keeps track of migrated files in the *daemon database*. The key to each file is its *bit-file identifier (BFID)*. For each migrated file, the daemon assigns a BFID that is stored in the file's inode. There is a daemon database record for each copy of a migrated file.

The daemon database also contains information such as the following:

- The MSP/VG name
- The MSP/VG key for each copy of a migrated file

When you use an MSP, the daemon database contains all of the information required to track a migrated file.

If you use an LS, there is also the *LS database*, which contains two tables of records:

- *Catalog (CAT) records* track the location of migrated data on volumes. There is one CAT record for each migrated copy of a file. If a migrated copy is divided between multiple volumes, there will be a CAT record for each portion or *chunk*.
- *Volume (VOL) records* contain information about the volumes. There is one VOL record for each volume.

Detailed information about the daemon and LS databases and their associated utilities is provided in "CAT Records" on page 536 and "VOL Records" on page 537.

Note: The databases consist of multiple files. However, these are not text files and cannot be updated by standard utility programs. See "Database Backups" on page 647.

There are also databases for DMF Manager performance records and alerts.

For information about the OpenVault database, see *OpenVault Administrator Guide for SGI InfiniteStorage*.

Ensuring Data Integrity

DMF software provides capabilities to ensure the integrity of offline data. For example, you can have multiple MSPs/VGs with each managing its own pool of volumes. Therefore, you can configure the DMF environment to copy filesystem data to multiple offline locations.

DMF software stores data that originates in a CXFS or XFS filesystem. Each object stored corresponds to a file in the native filesystem. When a user deletes a file, the inode for that file is removed from the filesystem. Deleting a file that has been migrated begins the process of invalidating the offline image of that file. In the LS, this eventually creates a gap in the volume. To ensure effective use of media, the LS provides a mechanism for reclaiming space lost to invalid data. This process is called *volume merging*.

Much of the work done by DMF software involves transaction processing that is recorded in databases. The DMF databases provide for full transaction journaling and employ two-phase commit technology. The combination of these two features ensures that DMF software applies only whole transactions to its databases. Additionally, in the event of an unscheduled system interrupt, it is always possible to replay the database journals in order to restore consistency between the DMF databases and the filesystem. DMF utilities also allow you to verify the general integrity of the DMF databases themselves. See "Administration Tasks" on page 51 for more information.

DMF Architecture

DMF software consists of the DMF daemon and one or more MSPs or LSs. The DMF daemon accepts requests to migrate filesystem data from the DMF administrator or from users. It also communicates with the operating system kernel to maintain a file's migration state in that file's inode.

The DMF daemon is responsible for dispensing a unique bit-file identifier (BFID) for each file that is migrated. The daemon also determines the destination of migration data and forms requests to the appropriate MSP/LS to make offline copies.

The MSP/LS accepts requests from the DMF daemon. For outbound data, the LS accrues requests until the amount of data justifies a volume mount. Requests for data retrieval are satisfied as they arrive. When multiple retrieval requests involve the same volume, all file data is retrieved in a single pass across the volume.

DMF software uses the DMAPI kernel interface defined by the Data Management Interface Group (DMIG). DMAPI is also supported by X/Open, where it is known as the *XDSM standard.*

Figure 1-16 illustrates the basic DMF architecture. Figure 1-17 shows the architecture of the LS.

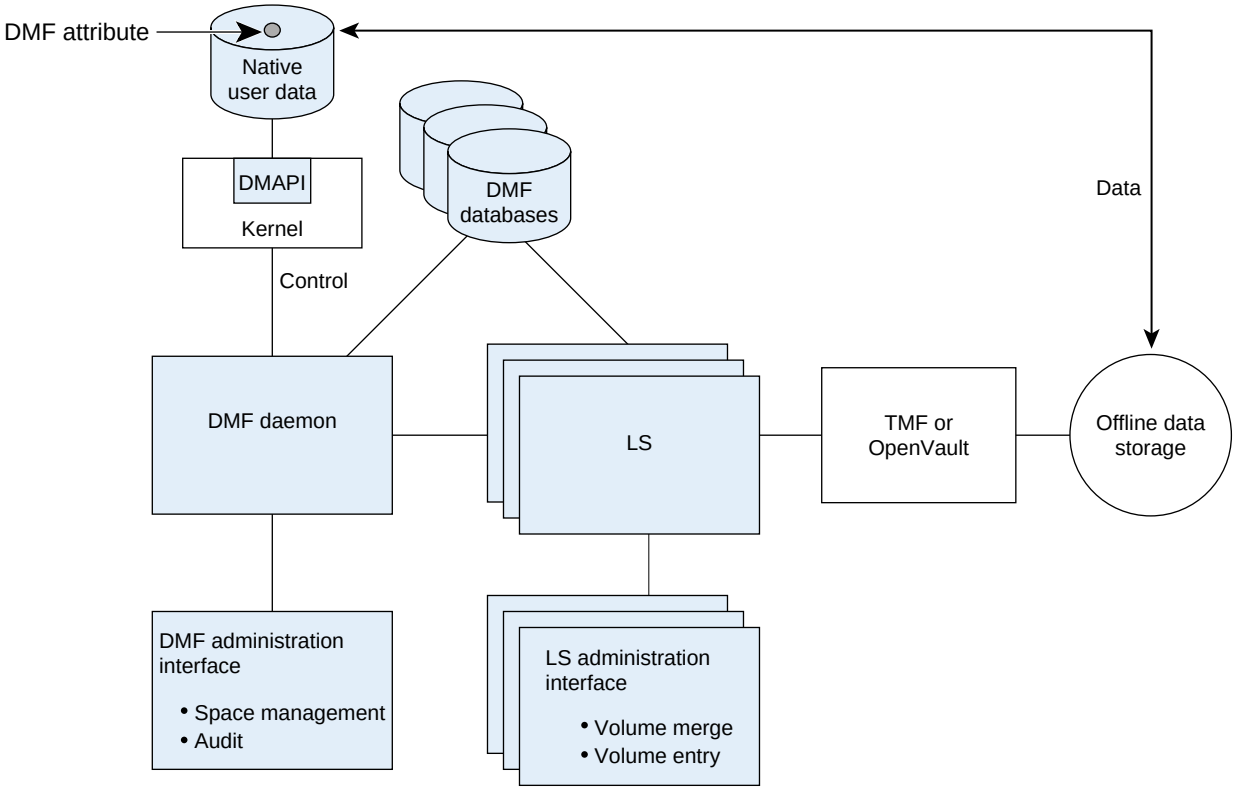

Figure 1-16 Basic DMF Architecture

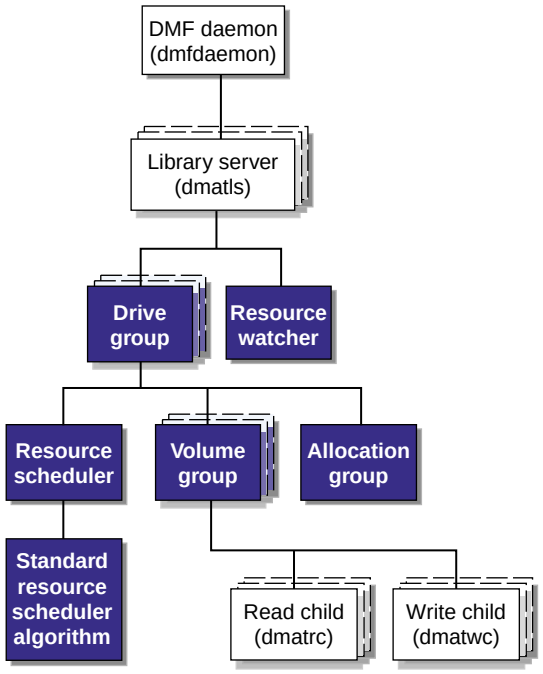

Figure 1-17 LS Architecture

There is one LS process (`dmatls`) per library, which maintains a database that all of its components share. The entities in the shaded boxes in Figure 1-17 are internal components of the `dmatls` process. Their functions are as follows:

Drive group (DG)

The DG is responsible for the management of a group of interchangeable drives located in the library. These drives can be used by multiple VGs (see *volume group* below) and by other processes, such as backups and interactive users. However, in the latter cases, the DG has no management involvement; the mounting service (TMF or OpenVault) is responsible for ensuring that these possibly competing uses of the drives do not interfere with each other.

	The main tasks of the DG are to: • Monitor I/O for errors • Attempt to classify the errors as volume, drive, or mounting service problems • Take preventive action
Volume group (VG)	The VG holds at most one copy of a migrated file in a pool of volumes, of which it has exclusive use. It can use only the drives managed by a single DG.
Allocation group (AG)	An AG is a pool of volumes that are transferred to a VG as needed and are returned to the pool when empty, subject to VG configuration parameters. Normally, an AG is configured to serve multiple VGs. Use of an AG is optional. When empty volumes are added to the DMF environment, they may be assigned to an AG via the `dmvoladm`(8) command.
Resource scheduler	In a busy environment, it is common for the number of drives requested by VGs to exceed the number available. The purpose of the resource scheduler is to decide which VGs should have first access to drives as they become available and to advise the DG of the result. The DMF administrator can configure the resource scheduler to meet site requirements.
Standard resource scheduler algorithm	This routine is an internal component of the `dmatls` process. Standard algorithms are provided with DMF software.
Resource watcher	The resource watcher monitors the activity of the other components and frequently updates files that contain data of use to the administrator. These are usually HTML files viewable by a web browser, but can also be text files designed for use by `awk` or `perl` scripts.

The `dmatrc` and `dmatwc` processes are called the *read children* and *write children*. They are created by VGs to perform the actual reading and writing of volumes. Unlike most of the other DMF processes that run indefinitely, these processes are created as needed, and are terminated when their specific work has been completed.

Media transports and robotic automounters are also key components of all DMF installations. Generally, DMF software can be used with any transport and automounter that is supported by either OpenVault or TMF. Additionally, DMF software supports *absolute block positioning*, a media transport capability that allows rapid positioning to an absolute block address on the volume. When this capability is provided by the transport, positioning speed is often three times faster than that obtained when reading the volume to the specified position.

Migrate Groups

A *migrate group (MG)* is a logical collection of MSPs and VGs that you combine into a set in order to have a single destination for a migrate request. A migration request to the MG will result in the copying of the file to exactly one MSP/VG that is a member of the MG.

You define an MG by adding the `migrategroup` object to the DMF configuration file. You can use the defined name of the MG in DMF policies and commands, similar to the way in which you use the names of VGs/MSPs. See:

- "Use Migrate Groups Appropriately" on page 111
- "Balance Data Among Libraries" on page 139
- "`migrategroup` Object" on page 426

DMF Capacity

The capacity of the DMF environment is measured in several ways, as follows:

- Total number of files. The daemon database can contain approximately 4 billion entries, and there is one database entry for each copy of a file that DMF software manages. Therefore, if there are two copies of each managed file, DMF software can theoretically manage approximately 2 billion files. The number of files that can be supported with best performance will vary depending upon the workload.
- Total amount of data. The capacity is limited only by the amount of secondary storage available to DMF software.
- Total amount of data moved between online and offline media. The number of drives configured for the DMF environment, the number of tape channels, and the number of disk channels all figure highly in the effective bandwidth. In general, DMF software provides full-channel performance to both tape and disk.

- File size. DMF software can support any file that can be created on the CXFS or XFS filesystem being managed.

DMF software has evolved in production-oriented, customer environments. It is designed to make full use of parallel and asynchronous operations, and to consume minimal system overhead while it executes, even in busy environments in which files are constantly moving online or offline. Exceptions to this rule will occasionally occur during infrequent maintenance operations when a full scan of filesystems or databases is performed.

For information about the DMF capacity license, see Chapter 2, "DMF Licensing" on page 67.

Requirements

Note: See the InfiniteStorage Software Platform (ISSP) release note and the DMF release note for the supported kernels, update levels, service pack levels, software versions, libraries, and tape devices.

This section discusses the following:

- "DMF Administrative and Store Directories" on page 44
- "PostgreSQL Database Server Requirements" on page 47
- "Server Node Requirements" on page 48
- "Parallel Data-Mover Node Requirements" on page 48
- "Mounting Service Requirements" on page 48
- "License Requirements" on page 48
- "DMAPI Requirement" on page 48
- "SAN Switch Zoning or Separate SAN Fabric Requirement" on page 49
- "DMF Manager Requirements" on page 49
- "DMF SOAP Requirements" on page 50
- "DMF Direct Archiving Requirements" on page 50

- "Fast-Mount Cache Requirements" on page 50
- "Cloud Storage Requirements" on page 51
- "Mediaflux Environment Requirements" on page 51

DMF Administrative and Store Directories

The DMF server uses the *DMF administrative and store directories* to store its databases, log files, journal files, and temporary files. Table 1-1 summaries the configuration parameters used to define these directories, the variable that represents the value of the parameter in this guide, and the purpose of the directories. For configuration details, see Chapter 7, "DMF Configuration File" on page 271.

Table 1-1 DMF Administrative and Store Directories

Configuration Parameter	Variable that Represents the Parameter Value	Example Value	Purpose
HOME_DIR	*HOME_DIR*	/dmf/home	Specifies the base pathname for directories in which the DMF daemon database, library server (LS) database, and related files reside. Minimum permission requirement: 711
SPOOL_DIR	*SPOOL_DIR*	/dmf/spool	Specifies the base pathname for directories in which DMF log files are kept. Minimum permission requirement: 711
JOURNAL_DIR	*JOURNAL_DIR*	/dmf/journals	Specifies the base pathname for directories in which the journal files for the daemon database and LS database will be written. Minimum permission requirement: 711
TMP_DIR	*TMP_DIR*	/dmf/tmp	Specifies the base pathname for directories in which DMF puts temporary files for its own internal use. Minimum permission requirement: 711
DATABASE_COPIES	*DATABASE_COPIES*	/dir1/database_copies and /dir2/database_copies	Specifies one or more directories into which the run_copy_databases.sh task will place a copy of the DMF databases.

Configuration Parameter	Variable that Represents the Parameter Value	Example Value	Purpose
DUMP_DESTINATION for integrated backups	*DUMP_DESTINATION*	/dmf/backups	Specifies the location in which to store backups and must be a dedicated filesystem mount point
DUMP_DESTINATION for disk backups (nonintegrated)	*DUMP_DESTINATION*	/dmf/backups	Specifies the location in which to store backups.
STORE_DIRECTORY for a DCM MSP	*STORE_DIRECTORY*	/dmf/*dcmmspname*_store	Specifies the directory that is holds files for a DCM MSP. There is one STORE_DIRECTORY parameter for each DCM MSP.
STORE_DIRECTORY for a disk MSP	*STORE_DIRECTORY*	/dmf/*diskmspname*_store	Specifies the directory that holds files for a disk MSP (there is one STORE_DIRECTORY parameter for each disk MSP).
CACHE_DIR	*CACHE_DIR*	/dmf/cache	*(Optional)* Specifies the directory in which the VG stores chunks while merging them from sparse volumes.
MOVE_FS	*MOVE_FS*	/move	*(Optional)* Specifies one or more scratch directories that are used by dmmove(8) to move files between media-specific processes (MSPs) or volume groups (VGs). You must specify a value for MOVE_FS if you intend to use the dmmove command. The best practice when using *MOVE_FS* is for it to be dedicated to the dmmove function.

For more information about DMF administrative directories, see "DMF Administrative and Store Directories" on page 44.

PostgreSQL Database Server Requirements

The PostgreSQL database is required by DMF for various purposes, including the queue-viewing tools:

- DMF requires the 9.3.*X* version of PostgreSQL. See "Apply Appropriate PostgreSQL Updates" on page 136.
- The PostgreSQL database server is the DMF server
- The PostgreSQL database server depends upon the `ident` service. See "Overview of the Installation and Configuration Steps" on page 165.
- The PostgreSQL database server runs as the `postgres` user. The `postgres` user must have at least `711` permission to the following directories:
  - *HOME_DIR*`/pg_data`
  - *JOURNAL_DIR*`/pg_xlog`
  - *TMP_DIR*`/pg_tmp`
  - *SPOOL_DIR*`/pglogs`
- The firewall must allow access to port 5432 for `localhost`
- SLES: The `postgres` user must be set to use a shell in `/etc/password`. (By default, the `postgres` user's shell is set to `/bin/false`.)

For configuration parameters specific to Postgres, see "`base` Object Parameters" on page 278.

For more information about DMF administrative directories, see "DMF Administrative and Store Directories" on page 44.

Server Node Requirements

A DMF server node requires the following:

- SGI x86_64 hardware
- One of the following operating systems as documented in the ISSP release note:
  - Red Hat Enterprise Linux (RHEL)
  - SUSE Linux Enterprise Server (SLES)
- DMF server software and associated products distributed with the ISSP release

Parallel Data-Mover Node Requirements

DMF parallel data-mover nodes require the following:

- SGI x86_64 hardware
- Same operating system as the DMF server and CXFS metadata server
- DMF parallel data-mover node software (which includes the required underlying CXFS client-only software)

If you use Parallel DMF, you must use OpenVault for those DGs that contain drives on parallel data-mover nodes. See "Parallel DMF Overview" on page 33.

Mounting Service Requirements

OpenVault requires `ksh`, not `pdksh`.

TMF has no requirements specific to DMF software.

License Requirements

DMF software is licensed. See Chapter 2, "DMF Licensing" on page 67.

DMAPI Requirement

For filesystems to be managed by DMF software, they must be mounted with the DMAPI interface enabled. See "Install DMAPI" on page 82.

SAN Switch Zoning or Separate SAN Fabric Requirement

Drives must be visible only from the active DMF server, the passive DMF server (if applicable), and the parallel data-mover nodes. The drives must not be visible to any other nodes. You must use one of the following:

- Independent switches (in a separate SAN fabric)
- Independent switch zones for CXFS/XVM volume paths and DMF drive paths

Warning: If the drives are visible to any other nodes, such as CXFS client-only nodes (other than those that are dedicated to being parallel data-mover nodes), data can become corrupted or overwritten.

DMF software requires independent paths to drives so that they are not fenced by CXFS. The ports for the drive paths on the switch must be masked from fencing in a CXFS configuration.

XVM must not fail over CXFS filesystem I/O to the paths visible through the tape/disk HBA ports when Fibre Channel port fencing occurs.

DMF Manager Requirements

DMF Manager has the following requirements:

- The DMF Manager software is installed on the DMF server node.
- One of the following web browsers:
  - Firefox 3.6 and later *(Firefox is the preferred browser)*
  - Internet Explorer versions supported under Windows 7 (ensure that the latest security patches are installed)

  Note: DMF Manager might also work other browsers, but its functionality is not tested.

- Before saving or applying configuration changes, you must make and mount the filesystems used for the DMF administrative directories. See "Configure Filesystems and Directories Appropriately for DMF" on page 94.

DMF SOAP Requirements

To use the DMF SOAP service capability, the software must be installed on the DMF server node.

DMF Direct Archiving Requirements

DMF direct archiving has the following requirements:

- The archive filesystem must be visible and mounted in the same location on the DMF server and any DMF parallel data-mover nodes. (The DMF server need not be the server of the archive filesystem; for example, the DMF server need not be the Lustre server.)
- The archive filesystem must be visible to DMF clients from which you want to run the `dmarchive`(1) command, but may have the filesystem mounted on a different mount point.
- The archive filesystem must be mounted on the DMF server and any DMF parallel data-mover nodes so that the `root` user is able to access the filesystem with `root` privileges (that is, with `root` *squashing* disabled).
- The archive filesystem must be fast enough to permit efficient streaming to/from secondary storage. If this is not the case, the speed could be so slow as to render DMF software useless; in that situation, copying the file to a managed filesystem via `cp`(1) and migrating the file may be a better option.

If a filesystem does not meet these requirements, do not add it to the DMF configuration file as an archive filesystem.

Fast-Mount Cache Requirements

The fast-mount cache feature requires the following at a minimum:

- Migrating at least two copies simultaneously, one copy to the cache (such as COPAN MAID) and at least one copy to a secondary-storage target (such as physical tape).
- Configuring a task to empty the cache.

However, SGI always recommends that you migrate at least two copies to secondary-storage targets in order to prevent file data loss in the event that a migrated

copy is damaged. When using a fast-mount cache, SGI therefore recommends that you migrate at least three copies (one to the cache and two to secondary-storage targets).

See "Use Fast-Mount Cache Appropriately" on page 112.

Cloud Storage Requirements

DMF software supports the following cloud systems as secondary storage:

- Scality RING private cloud
- Amazon Simple Storage Service (S3) public cloud

  Note: Amazon Glacier is not supported.

- Other products that present a service interface that is compatible with S3, in a private cloud

SGI strongly recommends that you migrate at least two copies to secondary-storage targets in order to prevent file data loss in the event that a migrated copy is damaged. A given cloud can be a single point of failure, therefore redundant copies within one cloud do not sufficiently protect against data loss. SGI therefore highly recommends that you migrate data to a second location (to another cloud instance, tape, or disk).

Mediaflux Environment Requirements

See Arcitecta's Mediaflux documentation for information about using DMF in a Mediaflux environment. In particular, there are certain requirements for configuring DMF and for using `dmaudit`(8) when restoring files in a managed filesystem.

Administration Tasks

This section discusses the following aspects of DMF administration:

- "Initial Planning" on page 52
- "Installation and Configuration" on page 52
- "Recurring Administrative Duties" on page 53

- "Commands Overview" on page 56

Initial Planning

DMF software manages two primary resources:

- Free space on managed filesystems
- Pools of secondary-storage media

You can configure those resources in a variety of environments, including the following:

- Support of interactive processing in a general-purpose environment with limited disk space
- Dedicated fileservers
- Lights-out operations

You must do the following:

- Evaluate the environment in which DMF software will run.
- Plan for a certain capacity, both in the number of files and in the amount of data
- Estimate the rate at which you will be moving data between the DMF store of data and the native filesystem
- Select autoloaders and media transports that are suitable for the data volume and delivery rates you anticipate

Installation and Configuration

You will install the DMF server software (which includes the software for TMF and OpenVault) from the ISSP media.

To configure the DMF environment, you must define a set of parameters in the DMF configuration file, typically by using a sample file as a starting point. See:

- "Best Practices for Configuration" on page 87
- Chapter 5, "Installing and Configuring the DMF Environment" on page 165

To make site-specific modifications, see "Customizing DMF" on page 186.

For a detailed example of configuring using COPAN cabinets, see:

- *COPAN MAID for DMF Quick Start Guide*
- *SGI 400 VTL for DMF Quick Start Guide*

Recurring Administrative Duties

DMF software requires that you perform recurring administrative duties in the following areas:

- "Free-Space Management" on page 53
- "File Ranking" on page 53
- "Offline Data Management" on page 54
- "Data Integrity and Reliability" on page 54

Note: You can use tasks that automate these duties. A *task* is a process initiated on a time schedule that you determine, similar to a `cron`(1) job. Tasks are defined with configuration file parameters and are described in detail in "`taskgroup` Object" on page 304 and "LS Tasks" on page 441.

Free-Space Management

You must decide how much free space to maintain on each managed filesystem. DMF software has the ability to monitor filesystem capacity and to initiate file migration and the freeing of space when free space falls below the prescribed thresholds. See Chapter 11, "Automated Space Management" on page 509.

File Ranking

You must decide which files are most important as migration candidates. When DMF software migrates and frees files, it selects files based on criteria you chose. The ordered list of files is called the *candidate list*. Whenever DMF software responds to a critical space threshold, it builds a new migration candidate list for the filesystem that reached the threshold. See "Generating the Candidate List" on page 510.

Offline Data Management

DMF software offers the ability to migrate data to multiple locations. Each location is managed by a separate MSP/VG and is usually constrained to a specific type of medium.

Complex strategies are possible when using multiple MSPs, LSs, or VGs. For example, short files can be migrated to a device with rapid mount times, while long files can be routed to a device with extremely high density.

You can describe criteria for MSP/VG selection. When setting up a VG, you assign a pool of volumes for use by that VG. The `dmvoladm`(8) utility provides management of the VG media pools.

You can configure DMF software to automatically merge volumes that are becoming sparse. With this configuration (using the `run_merge_tapes.sh` task for either disk or tape), the media pool is merged on a regular basis in order to reclaim unusable space.

Recording media eventually becomes unreliable. Sometimes, media transports become misaligned so that a volume written on one cannot be read from another. The following utilities support management of failing media:

- `dmatread`(8) recovers data
- `dmatsnf`(8) verifies LS volume integrity

Additionally, the volume merge process built into the LS is capable of effectively recovering data from failed media.

Chapter 14, "Library Servers and Media-Specific Processes" on page 531, provides more information on administration.

Data Integrity and Reliability

This section discusses the following things that you must do maintain the integrity and reliability of data managed by DMF software:

- "Run Backups" on page 55
- "Audit the Databases and Log files" on page 55
- "Protect Databases from Loss" on page 55
- "Remove Old Logs and Journals" on page 56

- "Hard-Delete Database Entries" on page 56

Run Backups

DMF software moves only the data associated with files, not the file inodes or directories, so you must still run filesystem backups in order to preserve the metadata associated with migrated files and their directories. You can configure DMF software to automatically run backups of your managed filesystems. See "Back Up Migrated Filesystems and DMF Databases" on page 137.

The `xfsdump`(8) and `xfsrestore`(8) utilities are aware of migrated files. The `xfsdump` utility can be configured to dump the data blocks for a file only if it has not yet been migrated. Files that are dual-state, partial-state, or offline have only their inodes backed up.

You can establish a policy of migrating 100% of the files in the managed filesystems before starting a backup, thereby leaving only a small amount of data that must be dumped. This practice can greatly increase the availability of the machine on which DMF software is running because, generally, backup commands must be executed in a quiet environment.

You can configure the `run_full_dump.sh` and `run_partial_dump.sh` tasks to ensure that all files have been migrated. These tasks can be configured to run when the environment is quiet.

See Chapter 4, "Backups and DMF" on page 141.

Audit the Databases and Log files

Configure DMF software to automatically run `dmaudit` to examine the consistency and integrity of the databases it uses. DMF databases record all information about stored data. The DMF databases must be synchronized with the filesystems that DMF software manages. Much of the work done by DMF software ensures that the DMF databases remain aligned with the filesystems.

Protect Databases from Loss

You can configure DMF software to periodically copy the databases to other devices on the system to protect them from loss (using the `run_copy_databases.sh` task). This task also uses the `dmdbcheck` utility to ensure the integrity of the databases before saving them.

Remove Old Logs and Journals

DMF software uses journal files to record database transactions. Journals can be replayed in the event of an unscheduled system interrupt that causes database corruption. You must ensure that journals are retained in a safe place until a full backup of the DMF databases can be performed.

You can configure the `run_remove_logs.sh` and `run_remove_journals.sh` tasks to automatically remove old logs and journals, which will prevent the DMF `SPOOL_DIR` and `JOURNAL_DIR` directories from overflowing.

Hard-Delete Database Entries

You can configure the `run_hard_deletes.sh` task to automatically remove database entries whose files will never be restored from backup media. See "Cleaning Up Obsolete Database Entries" on page 586.

Commands Overview

The DMF administrator has access to a wide variety of commands for controlling the DMF environment. This section discusses the following:

- "User Commands" on page 57
- "Licensing Commands" on page 58
- "Configuration Commands" on page 58
- "DMF Daemon and Related Commands" on page 59
- "Space Management Commands" on page 62
- "LS Commands" on page 63
- "DCM MSP Commands" on page 63
- "Disk MSP Command" on page 64
- "Other Commands" on page 64

Note: The functionality of some of these commands can be affected by site-defined policies; see "Customizing DMF" on page 186.

The FTP MSP uses no special commands, utilities, or databases.

User Commands

End users can run the following commands on DMF clients to affect the manual storing and retrieval of their data:

Command	Description
`dmarchive`(1)	Directly copies data between DMF secondary storage and a POSIX filesystem that is not managed by DMF software, such as Lustre. It is intended to streamline a work flow in which users work in an archive filesystem and later want to archive a copy of their data via DMF software. For more information about the `MIN_ARCHIVE_SIZE` parameter, see "`filesystem` Object Parameters" on page 351.
`dmattr`(1)	Displays whether files are migrated or not by returning a specified set of DMF attributes (for use in shell scripts).
`dmcapacity`(1)	Displays an estimate of the remaining storage capacity for each VG in each LS. You can optionally choose to report the data formatted into XML or HTML.
`dmcopy`(1)	Copies all or part of the data from a migrated file to an online file.
`dmdu`(1)	Displays the number of blocks contained in specified files and directories on a managed filesystem.
`dmfind`(1)	Displays whether files are migrated or not by searching through files in a directory hierarchy.
`dmget`(1)	Recalls the specified files.
`dmls`(1)	Displays whether files are migrated or not by listing the contents of a directory.
`dmoper`(1)	Displays outstanding requests for operator intervention.
`dmput`(1)	Migrates the specified files.
`dmtag`(1)	Allows a site-assigned 32-bit integer to be associated with a specific file (which can be tested in the `when`

	clause of particular configuration parameters and in site-defined policies).
`dmversion`(1)	Displays the version number of the currently installed DMF software.

The DMF `libdmfusr.so` user library lets you write your own site-defined DMF user commands that use the same application program interface (API) as the above DMF user commands. See Appendix B, "DMF User Library `libdmfusr.so`" on page 681.

Also see Chapter 16, "DMF SOAP Server" on page 597.

Licensing Commands

The following commands help you to manage DMF licenses:

Command	Description
`dmusage`(8)	Displays information about the capacity allowed by the DMF licenses and the amount of data that DMF software is currently managing against those licenses.
`dmflicense`(8)	Prints DMF license information.

Configuration Commands

The DMF configuration file (`/etc/dmf/dmf.conf`) contains *configuration objects* and associated *configuration parameters* that control the way DMF software operates. By changing the values associated with these objects and parameters, you can control the behavior of DMF software. To modify the configuration file, you can use DMF manager. For information about configuration, see:

- Chapter 5, "Installing and Configuring the DMF Environment" on page 165
- Chapter 6, "DMF Manager" on page 191
- Chapter 7, "DMF Configuration File" on page 271
- Chapter 8, "Parallel DMF Configuration" on page 485

The following man pages are also related to the configuration file:

Man page	Description
`dmf.conf`(5)	Describes the DMF configuration objects and parameters in detail.
`dmconfig`(8)	Prints DMF configuration parameters to standard output.

For detailed examples of configuring using COPAN cabinets, see:

- *COPAN MAID for DMF Quick Start Guide*
- *SGI 400 VTL for DMF Quick Start Guide*

DMF Daemon and Related Commands

The DMF daemon, `dmfdaemon`(8), communicates with the kernel through a device driver and receives backup and recall requests from users through a socket. The daemon activates the appropriate MSPs and LSs for file migration and recall, maintaining communication with them through unnamed pipes. It also changes the state of inodes as they pass through each phase of the migration and recall process. In addition, the daemon maintains a database containing entries for every migrated file on the system. Updates to database entries are logged in a journal file for recovery. See Chapter 12, "The DMF Daemon" on page 515, for a detailed description of the DMF daemon.

Caution: If used improperly, commands that make changes to the daemon database can cause data to be lost.

The following administrator commands are related to dmfdaemon and the daemon database:

Command	Description
dmaudit(8)	Reports discrepancies between filesystems and the daemon database. This command is executed automatically if you configure the run_audit.sh task.
dmcheck(8)	Checks the DMF installation and configuration and reports any problems.
dmdadm(8)	Performs daemon database administrative functions, such as viewing individual database records.
dmdbcheck(8)	Checks the consistency of a database by validating the location and key values associated with each record and key in the data and key files (also an LS command). If you configure the run_copy_database.sh task, this command is executed automatically as part of the task. The consistency check is completed before the DMF databases are saved. **Note:** See "Run Certain Commands Only on a Copy of the DMF Databases" on page 125.
dmdbrecover(8)	Applies journal records to a restored backup copy of the daemon database or LS database in order to create an up-to-date sane database.
dmdidle(8)	Causes files in pending requests to be flushed to secondary storage, even if this means forcing only a small amount of data to a volume.
dmdstat(8)	Indicates to the caller the current status of dmfdaemon.
dmdstop(8)	Stops the DMF daemon without consideration for related services.

	Note: To stop DMF, you should normally use the `service`(8) command. See "Starting and Stopping the DMF Environment" on page 181. For instructions about stopping DMF in an HA environment, see *High Availability Guide for SGI InfiniteStorage*.
`dmfdaemon`(8)	Starts the DMF daemon without consideration for related services.
	Note: To start DMF, you should normally use the `service`(8) command. See "Starting and Stopping the DMF Environment" on page 181. For instructions about starting DMF in an HA environment, see *High Availability Guide for SGI InfiniteStorage*.
`dmhdelete`(8)	Deletes expired daemon database entries and releases corresponding MSP/VG space, resulting in logically less active data. This command is executed automatically if you configure the `run_hard_deletes.sh` task.
`dmmigrate`(8)	Migrates regular files that match specified criteria in the specified filesystems, leaving them as dual-state. This utility is often used to migrate files before running backups of a filesystem, hence minimizing the size of the backup image. It may also be used in a DCM MSP environment to force cache files to be copied to secondary storage if necessary.
`dmsnap`(8)	Copies the daemon database and the LS database to a specified location. If you configure the

`run_copy_database.sh` task, this command is executed automatically as part of the task.

Space Management Commands

The following commands are associated with automated space management, which allows DMF software to maintain a specified level of free space on a filesystem through automatic file migration:

Command	Description
`dmfsfree`(8)	Attempts to bring the free space and migrated space of a filesystem into compliance with configured values.
`dmfsmon`(8)	Monitors the free space levels in filesystems configured with automated space management enabled (`auto`) and lets you maintain a specified level of free space.
`dmscanfs`(8)	Scans DMF filesystems or DCM MSP caches and prints status information to `stdout`.

See Chapter 11, "Automated Space Management" on page 509, for details.

LS Commands

The following commands manage the CAT and VOL records for the LS:

Command	Description
dmcatadm(8)	Provides maintenance and recovery services for the CAT records in the LS database.
dmvoladm(8)	Provides maintenance and recovery services for the VOL records in the LS database, including the selection of volumes for merge operations.

Most data transfers to and from secondary storage are performed by components internal to the LS. However, the following commands can read LS volumes directly:

Command	Description
dmatread(8)	Copies data directly from LS volumes to disk.
dmatsnf(8)	Audits and verifies the format of LS volumes.

The following commands check for inconsistencies in the LS database:

Command	Description
dmatvfy(8)	Verifies the contents of the LS database against the daemon database. This command is executed automatically if you configure the run_audit.sh task.
dmdbcheck(8)	Checks the consistency of a database by validating the location and key values associated with each record and key in the data and key files.

DCM MSP Commands

The following commands support the DCM MSP:

Command	Description
dmdskfree(8)	Manages file space within the disk cache and as needed migrates files to a lower tier and/or removes them from the disk cache.

`dmdskvfy`(8)	Verifies disk MSP file copies against the daemon database.

Disk MSP Command

The following command supports the disk MSP:

Command	Description
`dmdskvfy`(8)	Verifies disk MSP file copies against the daemon database.

Other Commands

The following commands are also available:

Command	Description
`dmcancel`(8)	Cancels some types of DMF requests by request ID.
`dmclripc`(8)	Frees system interprocess communication (IPC) resources and token files used by `dmlockmgr` and its clients when abnormal termination prevents orderly exit processing.
`dmcollect`(8)	Collects relevant details for problem analysis when DMF software is not functioning properly. You should run this command before submitting a bug report to SGI Support, should this ever be necessary.
`dmcopan`(8)	Provides detail about a COPAN MAID volume serial number (VSN) and its associated metadata.
`dmdate`(8)	Performs calculations on dates for administrative support scripts.
`dmdump`(8)	Creates a text copy of an inactive database file or a text copy of an inactive complete daemon database.
	Note: See "Run Certain Commands Only on a Copy of the DMF Databases" on page 125.
`dmdumpj`(8)	Creates a text copy of DMF journal transactions.

`dmfill`(8)	Recalls migrated files to fill a percentage of a filesystem. This command is mainly used in conjunction with backup and restore commands to return a corrupted filesystem to a previously known valid state.
`dmlockmgr`(8)	Invokes the database lock manager. The lock manager is an independent process that communicates with all applications that use the DMF databases, mediates record lock requests, and facilitates the automatic transaction recovery mechanism.
`dmmove`(8)	Moves copies of a migrated file's data to the specified MSPs/VGs.
`dmmvtree`(8)	Moves files from one managed filesystem to another without requiring that file data be recalled.
`dmov_keyfile`(8)	Creates the file of DMF OpenVault keys, ensuring that the contents of the file are semantically correct and have the correct file permissions. This command removes any DMF keys in the file for the OpenVault server system and adds new keys at the front of the file.
`dmov_loadtapes`(8)	Scans a library for volumes not imported into the OpenVault database and allows the user to select a portion of them to be used by a VG. The selected volumes are imported into the OpenVault database, assigned to the DMF application, and added to the LS database. This command can perform the equivalent actions for the filesystem backup scripts; just use the name of the associated task group instead of the name of a VG.
`dmov_makecarts`(8)	Makes the volumes in one or more LS databases accessible through OpenVault by importing into the OpenVault database any volumes unknown to it and by registering all volumes to the DMF application not yet so assigned. This command can perform the equivalent actions for the filesystem backup scripts; just use the name of the associated task group instead of the name of a VG.
`dmprojid`(8)	Sets or displays a file's site-defined project ID.

`dmqview`(8)	Display information about the internal DMF processing queues
`dmrepri`(8)	Reprioritizes recall and `dmcopy` request by daemon request ID.
`dmrestore`(8)	Restores DMF attributes to a file.
	Caution: This command is intended to be used in certain recovery scenarios; used incorrectly, it could cause DMF file-consistency issues.You should use this command only at the direction of SGI Support.
`dmselect`(8)	Selects migrated files based on given criteria. The output of this command can be used as input to `dmmove`(8).
`dmsort`(8)	Sorts files of blocked records.
`dmstat`(8)	Displays a variety of status information about the DMF environment, including details about the requests currently being processed by the daemon, statistics about requests that have been processed since the daemon last started, and details of current drive usage by VGs.
`dmtapestat`(8)	Displays drive metrics for the entire DMF installation. You execute this command as `root` from the DMF server.
`dmunput`(8)	Recalls files and removes them from DMF management. You execute this command as `root`.
`dmxfsrestore`(8)	Calls the `xfsrestore`(8) command to restore files backed up to volumes that were produced by DMF administrative maintenance scripts.
`sgi-ltfs`	Manages Linear Tape File System (LTFS) cartridges and drives.
`tsreport`(8)	Displays information about tape drive errors, alerts, and usage when the `ts` tape driver is used. The `tsreport` command is included in the `apd` RPM.

DMF Licensing

This chapter discusses the following:

- "DMF License Types" on page 67
- "Anticipating Your DMF Data Capacity Requirements" on page 69
- "Displaying Current DMF Data Capacity Use" on page 72
- "Parallel Data-Mover Option and Licensing" on page 74
- "Mounting Services and Licensing" on page 74
- "Gathering the Host Information" on page 74
- "Obtaining the License Keys" on page 75
- "Installing the License Keys" on page 75
- "Verifying the License Keys" on page 75
- "For More Information About Licensing" on page 79

DMF License Types

DMF uses software licensing based on SGI License Keys (LK). A production DMF environment requires that the following licenses are installed on the DMF server node: [1]

- *DMF server capability license.*
- One or more DMF *Parallel Data-Mover Option capability licenses* (if applicable).
- One or more cumulative *DMF data-capacity licenses* (*base* and optional *incremental*), available in different amounts, as shown in Table 2-1.

[1] To support training and functional demonstrations, DMF will run on a server with no license at all up to a maximum stored capacity of 1 TB without TMF or OpenVault.

At least one base data-capacity license is required. If multiple base data-capacity licenses are installed, they are additive.

In order to install an incremental data-capacity license, the total data capacity amount already installed (base plus incremental) must equal or exceed the amount of the new incremental amount. For example, to install a new 100TB+ incremental license, the environment must already be licensed for a total of 100 TB, which could be accomplished by several licensing methods, including any of the following:

– One 100TB base license

– One 10TB base license plus nine 10TB+ incremental licenses

– Two 10TB base licenses plus eight 10TB+ incremental licenses

Note: Some combinations are more cost-effective than others. For details about acquiring the proper set of licenses for your site, contact SGI Support.

Table 2-1 Data-Capacity License Amounts

Base Data-Capacity Amount	Incremental Data-Capacity Amount
10TB	10TB+
100TB	100TB+
1PB	1PB+
10PB	

In a high-availability (HA) environment, the passive DMF server requires the following licenses: a *DMF HA capability license* and a set of Parallel Data-Mover Option licenses and DMF data-capacity licenses equivalent to those on the active DMF server. For example, an HA DMF environment using two parallel data-mover nodes and an amount of managed data that requires two data-capacity licenses would require the following, as shown in Figure 2-1:

- Active DMF server:
  - 1 DMF server capability license
  - 2 Parallel Data mover Option capability licenses

  - 2 DMF data-capacity licenses
- Passive DMF server:
  - 1 DMF HA capability license
  - 2 Parallel Data-Mover Option capability licenses
  - 2 DMF data-capacity licenses

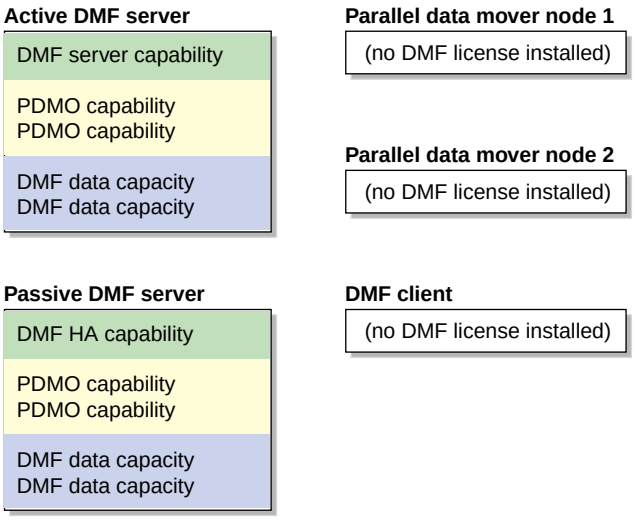

Figure 2-1 DMF Licenses

Anticipating Your DMF Data Capacity Requirements

DMF charges data against the license capacity when a file is scheduled for migration, either via a manual request (such as from a `dmput` command) or via an automated space-management operation (such as `dmfsfree`). The entire size of the file (including holes) is used in this calculation, not the amount of data in the file. DMF does not charge for files that are in regular state (`REG`).

Note: DMF only charges for the size of the original file, no matter how many copies are placed on DMF media; you can make as many copies as you like. The location of copies and their specific migration state do not affect the capacity license. For example, a dual-state file and an offline file are treated identically.

Therefore, you must install sufficient data-capacity licenses to cover the total size all of the files in all managed filesystems that have a state other than regular.

DMF credits data to the license capacity if the original file is deleted, modified, or removed from DMF management via a `dmunput`(1) command. The data in the DMF copies will be maintained under DMF management until the database entry for the file is hard-deleted, but its data will not count against the DMF license.

Figure 2-2 illustrates the concepts of the capacity license. In this example, 20 MB are charged against the license:

- FileA has a size of 10 MB, but it has 9 MB of data and a 1–MB hole. It has been migrated to three DMF copies in three VGs in a library server. Because the data still resides in the managed filesystem, the file is dual-state (`DUL`). The license is charged 10 MB for FileA.
- FileB has a size of 10 MB. It has been fully migrated to two copies, one on msp1 and another on msp2. No file data resides in the managed filesystem, so the file is in offline state (`OFL`). The license is charged 10 MB for FileB.
- FileC has a size of 10 MB, but because it is in regular state, it is not charged at all against the license.

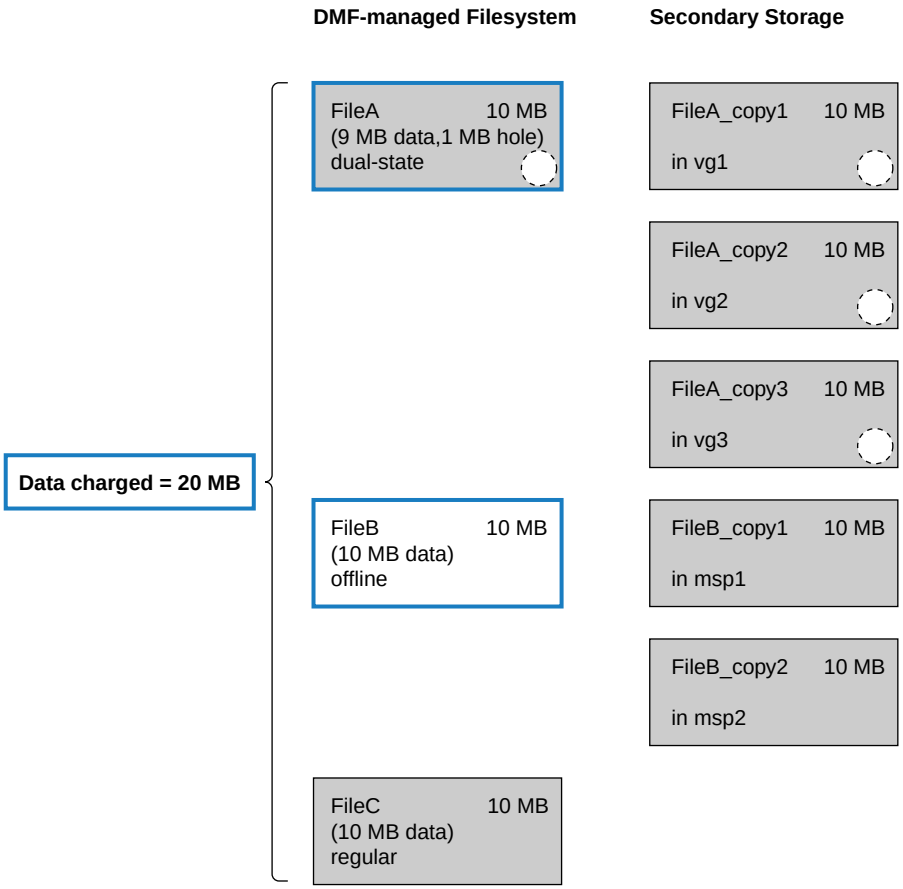

Figure 2-2 Data that Counts Towards the Capacity License

Displaying Current DMF Data Capacity Use

The dmusage(8) command or the **Show Usage** page in DMF Manager (see "Showing Current DMF Usage and Licensed Capacity" on page 205) will output information about the current DMF license capacity and the total number of bytes being managed by DMF.

There are two classes of DMF management displayed:

- **Data Under DMF Management** displays (for informational purposes) the total number of bytes that have been written to the following:
  - Library servers
  - Disk cache manager media-specific processes (DCM MSP)
  - Disk MSPs

  Note: Cloud MSP and FTP MSP copies are not included.

  These values are not the values charged against the DMF license.
- **Data Charged to DMF License** displays the total size of all existing files (including holes) in the managed filesystems that are under any state of DMF control, from migrating (MIG) to offline (OFL). Files in that are in regular (REG) state and the copies made by DMF **do not** apply against the license.

  These values will be smaller than the first class, because they do not include the following:
  - Copies made by DMF to any LS or MSP
  - Previously migrated files that have since been modified (and therefore placed back into regular state) or deleted. (This type of file is considered *soft-deleted* because its data might still reside on backup media. The file's data remains under DMF management until it has been hard-deleted, in case it is eventually restored. Soft-deleted files are not charged against the license.)

Note: The values displayed may be as much as 2 minutes out of date. Actions taken to delete files to get the filesystem usage back in compliance with the license will be acted on immediately by the daemon, but there may be a delay before they appear in the output of this command.

For example:

```
# dmusage

             Data Under DMF Management
        (does not include cloud or FTP MSPs)

   Store Type        Name           Bytes Managed
   ----------        ----           -------------
   Library Server    ls_copan       1015263000000 (1TB)
   Disk MSP          dcm                868163584 (868MB)
                                         --------
   Total bytes managed              1016131163584 (1TB)

        Data Charged to DMF License by Filesystem

   Mount Point                      Bytes Managed
   ----------                       -------------
   /dmiusr1                          511090800000 (511GB)
   /dmiusr2                          480055000000 (480GB)
   /dmiusr3                               9277697 (9MB)
   /dmiusr4                             646822912 (646MB)
                                         --------
   Total charged to license          991801900609 (991GB)
   DMF license capacity            10000000000000 (10TB)
   Percent of license capacity                  9
```

The DMF daemon compares the amount of data that DMF is currently managing against the licensed capacity and takes action if the following thresholds are exceeded:

- At 95%, the daemon will send a warning alert once per day.
- At 100%, the daemon will send a critical alert once per day. DMF will continue to function and will recall any data that has already been migrated, but further migrations will not be allowed. The daemon will check once every 2 minutes to see if the usage once again becomes legal (below capacity). This can be achieved by either of the following:
  - Deleting managed data
  - Adding one or more capacity licenses in order to increase the cumulative capacity total to the new desired limit.

Note: In order to install an incremental capacity license, the total capacity amount already installed (base plus incremental) must equal or exceed the amount of the new incremental amount. See "DMF License Types" on page 67.

The daemon will issue another alert when the usage once again becomes legal (below capacity).

Parallel Data-Mover Option and Licensing

Each *active parallel data-mover node* requires a corresponding license on the DMF server. DMF will allow as many DMF parallel data-mover nodes to become active at one time as there are DMF parallel data mover licenses in the DMF server's license file. (However, a parallel data mover license is not required for the DMF server's integrated data mover functionality.) No license is installed on the parallel data-mover node itself.

Mounting Services and Licensing

Use of the TMF or OpenVault mounting service requires DMF licenses.

Gathering the Host Information

When you order DMF, you will receive entitlement IDs for the licenses you purchased. You must submit the system host ID, host name, and entitlement IDs when requesting your permanent DMF license keys.

To obtain the host information for a server, view the **Licenses** panel in DMF Manager. See "Managing Licenses and Data Capacity with DMF Manager" on page 203.

You could also execute the following command:

```
/usr/sbin/lk_hostid
```

For example, the following shows that the serial number is `000423d5fd92` and the license ID is `23d5fd92`:

```
# /usr/sbin/lk_hostid
000423d5fd92 23d5fd92 socket=1 core=2 processor=2
#-------------------------------------------------------------------------
#The above is the default selected by lk_hostid. See below for additional
#hostid pairs.
#-------------------------------------------------------------------------
#Interface  SN                  LI         Driver ( Comment )
#-------------------------------------------------------------------------
 eth0       000423d5fd92        23d5fd92   e1000
 eth1       000423d5fd93        23d5fd93   e1000
```

Obtaining the License Keys

To obtain your DMF license keys, see information provided in your customer letter and the following web page:

http://www.sgi.com/support/licensing

Installing the License Keys

To install the license keys, copy them into the `/etc/lk/keys.dat` file or use the **Licenses** panel in DMF Manager to add the licenses (see "Adding New Licenses" on page 203).

Verifying the License Keys

You can verify your licenses in the following ways:

- "DMF Manager Licenses Panel" on page 76
- "`dmflicense`" on page 77
- "`lk_verify`" on page 77

DMF Manager Licenses Panel

You can view the **Licenses** panel in DMF Manager to determine the validity of the licenses, as shown in Figure 2-3. You must log in as the `Admin` user to DMF Manager in order to change licenses.

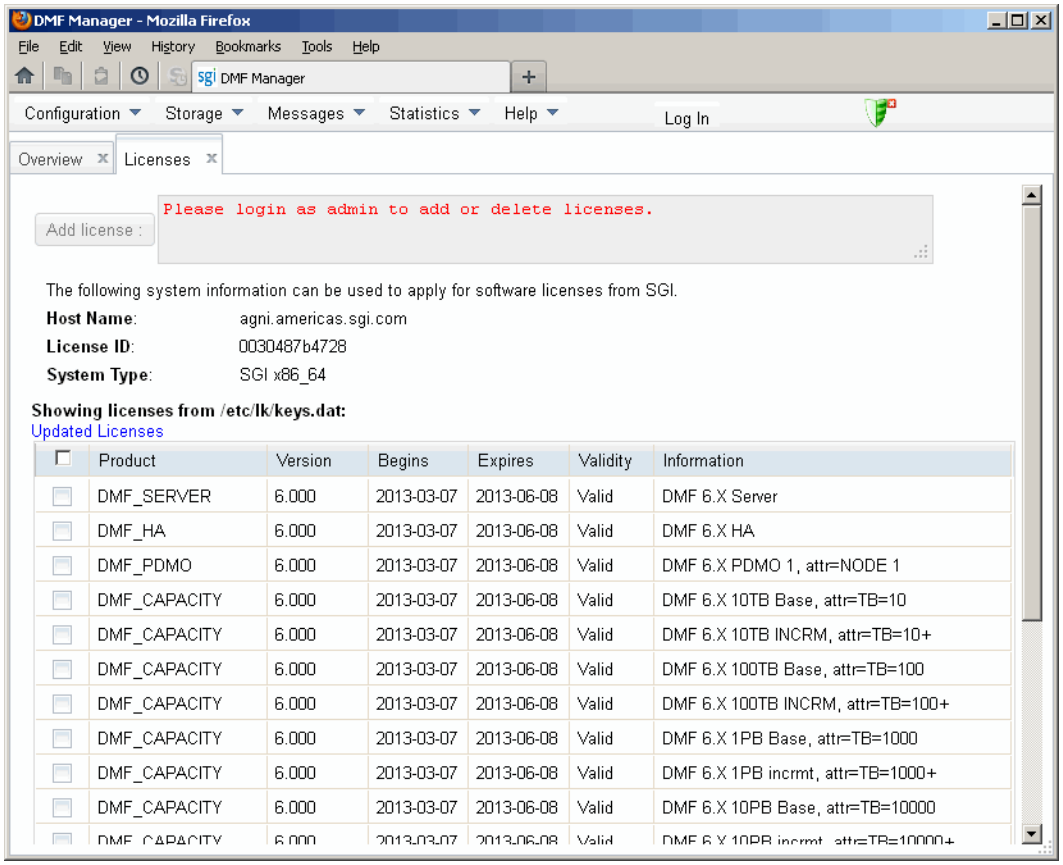

Figure 2-3 Licenses

For more information, see "Managing Licenses and Data Capacity with DMF Manager" on page 203.

dmflicense

You can use the dmflicense(8) command to verify the license keys. To see more output, use the -v option. For example:

```
# dmflicense -v
        File /etc/lk/keys.dat, line 6 is a valid DMF_SERVER license
        File /etc/lk/keys.dat, line 24 is a valid DMF_PDMO license
        File /etc/lk/keys.dat, line 29 is a valid DMF_PDMO license
        File /etc/lk/keys.dat, line 12 is a valid DMF_CAPACITY TB=100 license
        File /etc/lk/keys.dat, line 18 is a valid DMF_CAPACITY TB=100+ license
        Valid DMF license found.
        DMF capacity is 200TB.
```

lk_verify

You can use the lk_verify(1) command with the -A option to verify LK licenses. To see more output, use the -v option (you can use multiple times to display more output). For example:

```
# lk_verify -A -vvv
lk_check        All     All : total found=4

  1 /etc/lk/keys.dat:004        product=DMF_SERVER, version=6.000, count=0, begDate=1360172384, \
        expDate=0, licenseID=48bbb244, key=QPGc978utPAnG05MJPQ5l8sbjgSX3QE5, \
        info='DMF 6.X Server', vendor='Silicon Graphics International', \
        ref_id='270506'
               Verdict:         SUCCESS. Nodelock. Uncounted.
                                Available since today.
                                No End Date.

               Attribute 1 of 3 : info=DMF 6.X Server
               Attribute 2 of 3 : vendor=Silicon Graphics International
               Attribute 3 of 3 : ref_id=270506


  2 /etc/lk/keys.dat:009        product=DMF_HA, version=6.000, count=0, begDate=1360172520, \
        expDate=0, licenseID=48bbb244, key=CT7LtCI/C8vYc2JwS6k5BlYoeSVHDKsm, \
        info='DMF 6.X HA',vendor='Silicon Graphics International', ref_id='270507'
               Verdict:         SUCCESS. Nodelock. Uncounted.
                                Available since today.
```

```
                                No End Date.

                Attribute 1 of 3 : info=DMF 6.X HA
                Attribute 2 of 3 : vendor=Silicon Graphics International
                Attribute 3 of 3 : ref_id=270507


  3 /etc/lk/keys.dat:014        product=DMF_PDMO, version=6.000, count=0, begDate=1360172608, \
         expDate=0, licenseID=48bbb244, key=C8goMD0VwCtdIa8XIsbw94gidnYs+zIC, \
         info='DMF 6.X PDMO 1',attr='NODE 1', vendor='Silicon Graphics International', \
         ref_id='270508'
                Verdict:         SUCCESS. Nodelock. Uncounted.
                                 Available since today.
                                 No End Date.

                Attribute 1 of 4 : info=DMF 6.X PDMO 1
                Attribute 2 of 4 : attr=NODE 1
                Attribute 3 of 4 : vendor=Silicon Graphics International
                Attribute 4 of 4 : ref_id=270508


  4 /etc/lk/keys.dat:020        product=DMF_CAPACITY, version=6.000, count=0, begDate=1360172697, \
         expDate=0, licenseID=48bbb244, key=rn6Jiu3C2yZN8c0SNot5hq/1HSh6wuS9, \
         info='DMF 6.X 10TB Base',attr='TB=10', \
         vendor='Silicon Graphics International',ref_id='270509'
                Verdict:         SUCCESS. Nodelock. Uncounted.
                                 Available since today.
                                 No End Date.

                Attribute 1 of 4 : info=DMF 6.X 10TB Base
                Attribute 2 of 4 : attr=TB=10
                Attribute 3 of 4 : vendor=Silicon Graphics International
                Attribute 4 of 4 : ref_id=270509

lk_check        All     All : total matched=4
```

For More Information About Licensing

To request software keys or information about software licensing, see the following web page:

http://www.sgi.com/support/licensing

If you do not have access to the web, contact your local Customer Support Center.

Best Practices for DMF

This chapter discusses the following:

- "Best Practices for Installation, Upgrade, and Downgrade" on page 81
- "Best Practices for Configuration" on page 87
- "Best Practices for Administration" on page 123
- "Best Practices for Backups" on page 137
- "Best Practices for Optional Tasks" on page 138

Best Practices for Installation, Upgrade, and Downgrade

This section discusses the following:

- "Use the Correct Mix of Software Releases" on page 81
- "Install DMAPI" on page 82
- "Do Not Use YAST to Configure Network Services" on page 83
- "Upgrade Nodes in the Correct Order" on page 83
- "Take Appropriate Steps when Upgrading DMF" on page 84
- "Contact SGI Support to Downgrade After Using OpenVault™ 4.0 or Later" on page 87

Use the Correct Mix of Software Releases

In a production system, the active DMF server, the passive DMF server (in a high-availability environment), and any DMF parallel data-mover nodes should run the same versions of the following, as supported by a given InfiniteStorage Software Platform (ISSP) release:

- Operating system
- DMF

- CXFS (in a system with parallel data-mover nodes)

For details, see the ISSP release notes.

To support upgrading without having to take down the whole environment, nodes can temporarily run different releases during the upgrade process, as provided by the CXFS rolling upgrade procedure.

Caution: You must upgrade all CXFS server-capable administration nodes before upgrading any CXFS client-only nodes (including any parallel data-mover nodes, which are CXFS client-only nodes). Client-only nodes can temporarily run an earlier release than the server-capable administration nodes, during the upgrade process. Client-only nodes can never run a later release than the server-capable administration nodes.

Operating a cluster with client-only nodes running a mixture of older and newer CXFS versions may result in a performance loss. Relocation to a server-capable administration node that is running an older CXFS version is not supported.

Although CXFS client-only nodes (including DMF parallel data-mover nodes) that are not upgraded might continue to operate without problems, new functionality may not be enabled until all nodes are upgraded; SGI does not provide support for any problems encountered on the nodes that are not upgraded.

For details, see the section about CXFS release versions and rolling upgrades in the *CXFS 7 Administrator Guide for SGI InfiniteStorage.*

Install DMAPI

Data Management API (DMAPI) is the mechanism between the kernel and the XFS or CXFS filesystem for passing file management requests between the kernel and DMF. Ensure that you have installed DMAPI and the appropriate patches, and that you mount filesystems with the appropriate options. See "Use the Correct Mount Options" on page 99.

Do Not Use YAST to Configure Network Services

If you try to configure network services using YAST and you are using DHCP, YAST will modify the `/etc/hosts` file to include the following entry, where `hostname` is the name of your machine:

`127.0.0.2` *hostname hostname*

The above line will prevent `ov_admin`(8) from working because there cannot be multiple IP addresses defined for the DMF server hostname. You will see an error such as the following:

```
The OpenVault server name "hostname" matches this host's hostname,
but network packets for this hosts's IP address:
        127.0.0.2
are not being accepted by any installed ethernet card, so there appears
to be a problem with the configuration of /etc/hosts.  Please correct
this problem before continuing.
```

If you are using OpenVault, you should do one of the following:

- Remove the `127.0.0.2` line from the `/etc/hosts` file prior to configuring OpenVault
- Do not use YAST to configure network services

Upgrade Nodes in the Correct Order

You should upgrade nodes in the following order:

1. Passive DMF server (if using HA)
2. OpenVault server
3. Active DMF server
4. Parallel data-mover nodes (if used)
5. DMF clients

Take Appropriate Steps when Upgrading DMF

Note: If you are upgrading from DMF 3.9 or earlier, see the information about upgrade caveats in the ISSP release note for more information.

To perform an upgrade, do the following:

1. Read the ISSP release note, DMF release note, and any late-breaking caveats on the online download page. Pay particular attention to any installation and upgrade caveats.
2. Stop all applications that are writing data to the managed filesystems.
3. Save the established DMF and mounting service configurations to an external storage medium.
4. Ensure that DMF is stopped. In an HA environment, see *High Availability Guide for SGI InfiniteStorage*. In a non-HA environment, execute the following:

   ```
   dmfserver# service dmf stop
   ```

Caution: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage*.

5. Ensure that the applicable mounting service is stopped. In a non-HA environment, execute the following (assuming for simplicity that the DMF server is also the mounting-service server):
   - TMF:

     ```
     dmfserver# service tmf stop
     ```

   - OpenVault:

     ```
     dmfserver# service openvault stop
     ```

6. If the DMF administrative directories are in XFS filesystems, make a copy of the `fstab`(5) file. For example:

   ```
   dmfserver# cp /etc/fstab /myupgrade/fstab
   ```

7. Make a copy of the following:

   a. The DMF configuration file `dmf.conf`. For example:

      ```
      dmfserver# cp /etc/dmf/dmf.conf /myupgrade/dmf.conf
      ```

   b. The mounting service configuration information:

      - TMF: copy the `tmf.config` file to a safe location. For example:

        ```
        dmfserver# cp /etc/tmf/tmf.config /myupgrade/tmf.config
        ```

      - OpenVault (if the OpenVault configuration is set up on the boot partition and not under a DMF administrative directory): create a compressed file of the OpenVault configuration directory `/var/opt/openvault`. For example:

        ```
        # cd /var/opt
        # /bin/tar cf /myupgrade/somefile.tar openvault/*
        # /usr/bin/compress /myupgrade/somefile.tar
        ```

   c. Networking files for `exports`(5), `auto.master`(5), and `resolve.conf`(5). For example:

      ```
      # cp /etc/exports /myupgrade/exports
      # cp /etc/auto.master /myupgrade/auto.master
      # cp /etc/resolv.conf /myupgrade/resolv.conf
      ```

8. Upgrade the operating system software to the level supported by the version of DMF that you are upgrading to, paying particular attention to any installation and upgrade caveats in the release notes and any late-breaking caveats on the online download page.

9. If your DMF administrative directories are in XFS filesystems, do the following:

   Note: To avoid copying the `fstab` information from a previous partition, **do not copy** the saved `/myupgrade/fstab` file to the new `/etc` directory in the upgraded system.

   ---

   a. Use the `cat`(1) command to view the previous `fstab` file:

      ```
      # cat /myupgrade/fstab
      ```

The following is an example of how DMF administrative directories and user directories that are managed by DMF (in this case, `/dmfusr1` and `/dmfusr3`) could be set up within `/etc/fstab`:

```
/dev/lxvm/home          /dmf/home       xfs     defaults                0 0
/dev/lxvm/journals      /dmf/journals   xfs     defaults                0 0
/dev/lxvm/move          /move           xfs     dmi,mtpt=/move          0 0
/dev/lxvm/spool         /dmf/spool      xfs     defaults                0 0
/dev/lxvm/cache         /dmf/cache      xfs     dmi,mtpt=/dmf/cache     0 0
/dev/lxvm/tmp           /dmf/tmp        xfs     defaults                0 0
/dev/lxvm/dmfusr1       /dmfusr1        xfs     dmi,mtpt=/dmfusr1       0 0
/dev/lxvm/dmfusr3       /dmfusr3        xfs     dmi,mtpt=/dmfusr3       0 0
```

b. Verify the existence of the matching XFS devices on the upgraded system by using the `ls`(1) command:

```
# ls -al /dev/lxvm*
```

c. Copy and paste the DMF administrative directory entry lines (those that contain `/dmf/`*directoryname*) from the copy of the `fstab` file (`/myupgrade/fstab`) into the new `/etc/fstab` for the upgraded system.

10. Reestablish the files and directories copied in step 7 above to their normal locations on the upgrade system. For example:

```
# cp /myupgrade/dmf.conf /etc/dmf/dmf.conf
# cp /myupgrade/exports /etc/exports
# cp /myupgrade/auto.master /etc/auto.master
# cp /myupgrade/resolv.conf /etc/resolv.conf
```

If TMF, also:

```
# cp /myupgrade/tmf.config /etc/tmf/tmf.config
```

If OpenVault (and if the OpenVault configuration is set up on the boot partition and not under a DMF administrative directory), also do the following, for example:

```
# cd /var/opt
# /bin/tar xf /myupgrade/somefile.tar.Z
```

11. Follow upgrade instructions in the ISSP release note to update the DMF and mounting service software.

12. Run the dmcheck(8) command, which will identify any issues with using your existing DMF configuration file with the upgraded software.

Contact SGI Support to Downgrade After Using OpenVault™ 4.0 or Later

If you are running OpenVault and want to downgrade after using OpenVault 4.0 or later, you must contact SGI support for assistance.

Best Practices for Configuration

This section discusses the following:

- "Plan Ahead for Disaster Recovery" on page 88
- "Ensure Quick Communication Among Nodes in the DMF Environment" on page 91
- "Enable RSCN Suppression Appropriately" on page 93
- "Follow All DMF Requirements" on page 93
- "Use Supported Libraries and Tape Drives" on page 94
- "Use Sufficiently Fast Filesystems" on page 94
- "Configure Passwordless SSH" on page 94
- "Configure Filesystems and Directories Appropriately for DMF" on page 94
- "Migrate at Least Two Copies of a File" on page 100
- "Safely Make Changes to the DMF Configuration" on page 100
- "Use Inode-Resident Extended Attributes and 256–byte Inodes" on page 104
- "Limit Path Segment Extension Records" on page 104
- "Do Not Change Script Names" on page 104
- "Configure DMF Appropriately with CXFS™" on page 104
- "Improve Drive Performance with an Appropriate VG Zone Size" on page 105
- "Appropriately Force Small Files to Migrate" on page 107

- "Add HBA Drivers to the `initrd` Image" on page 107
- "Set `RECALL_NOTIFICATION_RATE` to `0` if CXFS Range Tokens are Disabled" on page 107
- "Set the `xinetd` `tcpmux` `instances` Parameter Appropriately" on page 108
- "Avoid Unintentional File Recall by Filesystem Browsers" on page 108
- "Configure Appropriately for SGI 400 VTL or COPAN MAID Shelves" on page 109
- "Use Migrate Groups Appropriately" on page 111
- "Use Fast-Mount Cache Appropriately" on page 112
- "Use Cloud Storage Appropriately" on page 114
- "Minimize Access Times by Using `NAME_FORMAT`" on page 116
- "Ensure that the Cache Copy is Recalled First" on page 116
- "Use a Task Group to Run `dmmigrate` Periodically" on page 116
- "Restrict the Size of the Alerts and Performance Records Databases" on page 117
- "Prevent Stalled-Recovery Timeout in a Non-HA Environment" on page 118
- "Use Appropriate Tape Barcodes" on page 119
- "Use `dmarchive` to Copy Archive File Data to Secondary Storage" on page 119
- "Use an Appropriate Filesystem for a Disk MSP" on page 121
- "Use Corresponding Drive-Group Names in OpenVault and DMF" on page 121
- "Use a Private Network Interface in a Parallel Environment" on page 121
- "Modify Partial-State Capability with Care" on page 122
- "Use the Correct Child-Process Constraints for a DCM MSP" on page 122
- "Explicitly Specify the Server on OpenVault Commands" on page 122

Plan Ahead for Disaster Recovery

This section discusses the following:

- "Best Practices to Permit Future Disaster Recovery" on page 89
- "When is Disaster Recovery Possible?" on page 89
- "What Must Be Protected" on page 89
- "Why Backups Are Needed" on page 90
- "Recovery Possibilities" on page 91
- "Best Practices for Performing Disaster Recovery" on page 91

Best Practices to Permit Future Disaster Recovery

The following are best practices that will permit disaster recovery if the need arises:

- Always make multiple copies of migrated data on separate media (see "Migrate at Least Two Copies of a File" on page 100)
- Use the DMF integrated backup method, which includes a disaster-recovery bundle that includes a backup filesystem with the DMF databases and administrative files (see "Best Practices for Backups" on page 137)
- Store backups and at least one data copy off-site to permit recovery in the case of a physical disaster, such as a fire

When is Disaster Recovery Possible?

Disaster recovery is possible if you are using the integrated-backup method. Careful planning and implementation will help you recover if a disaster occurs.

If the DMF database is lost or corrupt and the latest database backup file residing in the backup filesystem becomes inaccessible (either due to filesystem corruption or due the file being in an offline state), you must first restore access to the backup filesystem and subsequently to the DMF database in order to restore the server.

What Must Be Protected

The DMF system is composed of the following pieces:

- DMF databases and administrative files, which store all of the details about how and where file data was migrated to secondary storage. Each file is assigned a unique key, known as a BFID. All transactions committed to the databases are also stored in external journal files for a configurable period of time.

- Filesystem inodes and directory structures, which remain in the managed filesystem. DMF stores an extended attribute in each inode with information specific to DMF, including the BFID. When files are migrated, the data moves offline but the inode remains in the managed filesystem in order to provide persistent access to the filesystem namespace. Because XFS and CXFS are journaling filesystems, all metadata operations to the filesystem are also written to an internal log area.
- File data, which is migrated to secondary storage

All of the above must be present in order for the DMF system to be whole, as depicted in Figure 3-1.

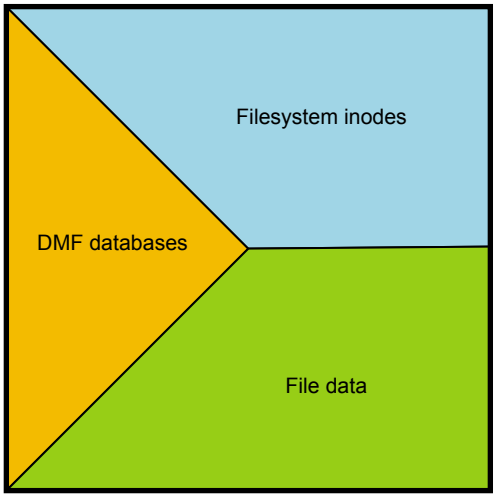

Figure 3-1 Components Required by the DMF System

Why Backups Are Needed

When using DMF to make multiple copies of migrated data, you must still perform regular backups in order to protect inodes, directory structures, and unmigrated files; DMF moves only the data associated with files, not the file inodes or directories.

The managed filesystems and the DMF databases should be backed up on a frequent and regular basis.

Recovery Possibilities

If one migrated copy is lost, data is available from another migrated copy.

If just the DMF database is lost, no transactions will be lost if the journals are applied to the last good database copy that can be restored from one of the backups.

If both the DMF database and the journals are lost, transactions after the last good dump will be lost; however, with some effort, some of these lost transactions might be reconstructed from the DMF logs and the inodes.

If the filesystem is corrupted, the dump files produced in the backup would be used to restore the inodes.

Best Practices for Performing Disaster Recovery

See "Disaster Recovery Procedures" on page 650.

Ensure Quick Communication Among Nodes in the DMF Environment

DMF must be able to quickly communicate directly between every potential DMF server node, parallel data-mover node, and potential OpenVault server node using IP addresses (both IPv4 and IPv6) and logical names, without routing. You must therefore ensure that all nodes have a consistent mapping of hostnames to IP addresses, using one of the following methods:

- "Preferred Method: Consistent `/etc/hosts` File on All Nodes and Access Local First" on page 91
- "Acceptable Method: Reliable DNS in a Dedicated HA cluster" on page 93

Each method has advantages and disadvantages, but the `/etc/hosts` method is preferred because it is the fasted and simplest method.

Note: For simplicity, SGI recommends that you avoid the Network Information Service (NIS).

Preferred Method: Consistent `/etc/hosts` File on All Nodes and Access Local First

A consistent and accurate `/etc/hosts` file provides the fastest look-up because it is cached in memory, and is therefore the preferred method to ensure quick

communication among cluster nodes. However, it requires a manual edit of static files that must be updated individually, and a given file can be accidentally corrupted.

To use the `/etc/hosts` method, do the following:

1. For each node (every potential DMF server node, parallel data-mover node, and potential OpenVault server node), decide if you want to specify the fully qualified domain name or the simple hostname. This decision usually depends upon how the clients are going to resolve names for their client/server programs (such as NFS), how their default resolution is done, and so on. If you use the fully qualified domain name for a particular node, then all of the nodes should use the fully qualified name of that node when defining the IP/hostname information for that node in their `/etc/hosts` files (IPv4 and IPv6).

2. Edit the `/etc/hosts` file so that it contains consistent entries for all nodes and their private interfaces as well. The file has the following format, where *primary_hostname* can be the simple hostname or the fully qualified domain name:

   IP_address primary_hostname aliases

   For example:

   ```
   10.0.3.1 priv-server1
   10.0.3.2 priv-server2
   10.0.3.3 priv-client1
   190.0.2.1 server1.example.com server1
   190.0.2.2 server2.example.com server2
   190.0.2.3 client1.example.com client1
   ```

3. Add all of these IP addresses to the `/etc/hosts` file on each node.

4. Edit the `/etc/nsswitch.conf` file so that local files are accessed before DNS (or NIS). That is, the `hosts` line in `/etc/nsswitch.conf` must list `files` first. For example:

   ```
   hosts:      files dns nis
   ```

   (The order of `dns` and `nis` is not significant to DMF, but `files` must be first.)

For more information, see the `hosts`(5) and `resolve.conf`(5) man pages.

Acceptable Method: Reliable DNS in a Dedicated HA cluster

The domain name service (DNS) lets you have a single authoritative source for mapping hostnames to IP addresses (IPv4 and IPv6) and avoids the need to manually edit and update static files on every node. However, it can reduce availability if the DNS service becomes unreliable. If you choose to use the DNS method, you must ensure that it is very reliable, which can involve more work.

To use the DNS method, do the following:

1. Use multiple DNS servers and set up a high-availability (HA) cluster specifically for the DNS IP addresses. Do not include this HA service within a DMF HA cluster; use a dedicated HA cluster.
2. Add the following line to the `/etc/resolv.conf` file so that the resolver will rotate to another DNS server if the current server does not respond within 1 second:

   ```
   options timeout:1 rotate
   ```

   Note: Having multiple DNS servers is not sufficient, even with the above rotation timeout. An HA cluster is required for reliability.

Enable RSCN Suppression Appropriately

See "Suppressing RSCN on Host Port HBAs" on page 177.

Follow All DMF Requirements

Ensure that you follow all of the requirements for DMF listed in "Requirements" on page 43:

- "Server Node Requirements" on page 48
- "Parallel Data-Mover Node Requirements" on page 48
- "Mounting Service Requirements" on page 48
- "License Requirements" on page 48
- "DMAPI Requirement" on page 48

- "SAN Switch Zoning or Separate SAN Fabric Requirement" on page 49
- "DMF Manager Requirements" on page 49
- "DMF SOAP Requirements" on page 50
- "DMF Direct Archiving Requirements" on page 50
- "Fast-Mount Cache Requirements" on page 50
- "Cloud Storage Requirements" on page 51

Use Supported Libraries and Tape Drives

For the list of supported TMF and OpenVault libraries and supported tape drives, see the DMF release notes.

Use Sufficiently Fast Filesystems

A filesystem on which DMF operates must be fast-enough to permit efficient streaming to/from all secondary storage media. This is particularly important for tape drives, because slow I/O can lead to increased wear on the drive and cartridges (due to excessive stopping and starting of the drive heads).

Configure Passwordless SSH

If you are running DMF in an HA environment or using the Parallel Data-Mover Option, you should configure passwordless secure shell (SSH) so that DMF can properly gather, distribute, and display information. See "Passwordless SSH Configuration for DMF" on page 176.

Configure Filesystems and Directories Appropriately for DMF

This section discusses the following:

- "Do Not Place Administrative and Store Directories in a Managed Filesystem" on page 95
- "Use Different Physical Devices for *HOME_DIR* and *JOURNAL_DIR*" on page 95
- "Avoid Root Filesystem Location for a Production System" on page 95

- "Restrict Administrative and Store Directories to DMF Use" on page 95
- "Use CXFS Filesystems with the Parallel Data-Mover Option" on page 96
- "Sizing Guidelines" on page 96
- "Use the Correct Mount Options" on page 99

Also see "DMF Administrative and Store Directories" on page 44.

Do Not Place Administrative and Store Directories in a Managed Filesystem

A DMF administrative or store directory must not be in filesystem that is managed by DMF.

Use Different Physical Devices for *HOME_DIR* and *JOURNAL_DIR*

To provide the best chance for database recovery, *HOME_DIR* must be on a different physical device from *JOURNAL_DIR*.

Avoid Root Filesystem Location for a Production System

By default, the DMF daemon requires that a DMF administrative directory does not reside in the root filesystem. For testing or demonstration purposes, you can override this requirement for all but *MOVE_FS* and a DCM MSP *STORE_DIRECTORY* by using the `ADMDIR_IN_ROOTFS` parameter; however, SGI does not recommend overriding the requirement for a production system. See "`base` Object Parameters" on page 278.

Restrict Administrative and Store Directories to DMF Use

You will place the DMF administrative and store directories on a general-purpose RAID storage system.

In a production system, SGI in most cases recommends that you restrict these directories to DMF use and make them the mount point of a filesystem, in order to limit the loss of data in the case of a filesystem failure.

The following are **required** to be dedicated to DMF use and to be a filesystem mount point:

MOVE_FS (if used)
DCM MSP *STORE_DIRECTORY*
DUMP_DESTINATION for integrated backups

SGI suggests that other directories also be used exclusively by DMF and to be a filesystem mountpoint, as shown in Table 3-1.

Table 3-1 DMF Exclusive Use of Filesystems

Required	Suggested
STORE_DIRECTORY for a DCM MSP	*STORE_DIRECTORY* for a disk MSP
DUMP_DESTINATION for integrated backups	*HOME_DIR*
MOVE_FS	*SPOOL_DIR*
	JOURNAL_DIR
	TMP_DIR
	DATABASE_COPIES
	CACHE_DIR (if used)

Use CXFS Filesystems with the Parallel Data-Mover Option

When using the Parallel Data-Mover Option, the following must be CXFS filesystems or be in CXFS filesystems:

HOME_DIR
SPOOL_DIR
TMP_DIR
MOVE_FS
CACHE_DIR (if used)
DCM MSP *STORE_DIRECTORY*

Sizing Guidelines

Note: You must evaluate these guidelines in terms of the specifics at your site, rounding up to allow margin for error.

The following sections provide guidelines for sizing the filesystems that DMF requires:

• "*HOME_DIR* Size" on page 97

- "*JOURNAL_DIR* Size" on page 98
- "*SPOOL_DIR* Size" on page 98
- "*TMP_DIR* Size" on page 98
- "*MOVE_FS* Performance and Size" on page 98

In general, these filesystems should be sized in terms of gigabytes. Table 3-2 shows the minimum recommended sizes.

Table 3-2 Minimum Sizes for DMF Directories

Directory	Minimum Recommended Size (GB)
HOME_DIR	500
JOURNAL_DIR	75
SPOOL_DIR	200
TMP_DIR	500
MOVE_FS	Capacity of one new volume

For individual guidelines and requirements for each directory, see the specific parameter descriptions in Chapter 7, "DMF Configuration File" on page 271.

See also "Safely Make Changes to the DMF Configuration" on page 100.

HOME_DIR Size

HOME_DIR will require approximately the following:

- The daemon and LS databases require approximately 500 MB per 1 million migrated files, per DMF copy. If you make two copies, they would require approximately 1 GB (that is, 500 MB x 2).
- An alerts database of 1 MB can hold approximately 5,400 records.
- A performance records database of 1 MB can hold approximately 5,130 records

You can purge old records after specified period of time. See "Restrict the Size of the Alerts and Performance Records Databases" on page 117.

Note: Other database information (such as the OpenVault server database in an HA configuration) requires an insignificant amount of space in comparison.

JOURNAL_DIR Size

JOURNAL_DIR will require approximately 500 MB per 1 million database operations (such as migrate, recall, and hard delete). You can set the `JOURNAL_RETENTION` parameter to purge old journals after a period of time. The absolute minimum value should be the time since the last successful backup of the DMF databases.

SPOOL_DIR Size

SPOOL_DIR will require approximately 1 MB per 500 DMF requests. You can set the `LOG_RETENTION` parameter to purge old logs after a period of time.

TMP_DIR Size

TMP_DIR is used for various temporary storage needs for DMF, such as the following:

- If you do not have a dedicated *CACHE_DIR* directory, cache merges will use *TMP_DIR*. The `libraryserver` object's `CACHE_SPACE` parameter controls how much space is used for cache merges.
- If backups are being done to tape, a temporary snapshot of the DMF databases is stored in *TMP_DIR* before being written to tape. (See `HOME_DIR` for database size.)
- The `run_filesystem_scan.sh` task places its output file in *TMP_DIR* by default. This file is approximately 150 MB for every 1 million files contained in the managed filesystems.

MOVE_FS Performance and Size

The *MOVE_FS* filesystem should have performance characteristics similar to the primary managed filesystems because DMF will follow the same rules for drive utilization as defined in the drive groups (DGs) and VGs (the `DRIVE_MAXIMUM` and `MAX_PUT_CHILDREN` parameters) when moving large numbers of files. A *MOVE_FS* filesystem with slower bandwidth than what the `DRIVE_MAXIMUM` and `MAX_PUT_CHILDREN` parameters are tuned for may become overloaded with DMF requests. In extreme cases, DMF can become backlogged on *MOVE_FS* and delay the processing of user requests.

The size of *MOVE_FS* should be approximately the capacity of a data cartridge, including compression, times the `MAX_PUT_CHILDREN` value.

For example:

*500 GB native capacity * 1.6 compression * 3 drives = 2.4 TB*

Use the Correct Mount Options

Table 3-3 on page 99 lists the appropriate mount options for managed filesystems and filesystems that contain DMF administrative directories and store directories.

Table 3-3 Required and Suggested Mount Options

Directory / Filesystem	Required Mount Options	Suggested Mount Options
Managed filesystems	`dmi,mtpt=`*mountpoint*	`logbufs=8,logbsize=256k`
HOME_DIR		`logbufs=8,logbsize=256k`
SPOOL_DIR		`logbufs=8,logbsize=256k`
JOURNAL_DIR		`logbufs=8,logbsize=256k`
TMP_DIR	`inode64`	`logbufs=8,logbsize=256k`
DATABASE_COPIES		`logbufs=8,logbsize=256k`
DUMP_DESTINATION for disk backups		`logbufs=8,logbsize=256k`
DUMP_DESTINATION for integrated backups	`dmi,mtpt=`*mountpoint*	`logbufs=8,logbsize=256k`
DCM MSP *STORE_DIRECTORY*	`dmi,mtpt=`*mountpoint*`,dirsync`	`logbufs=8,logbsize=256k`
Disk MSP *STORE_DIRECTORY*	`dirsync`	`logbufs=8,logbsize=256k`
CACHE_DIR	`inode64`	`logbufs=8,logbsize=256k`
MOVE_FS	`dmi,mtpt=`*mountpoint*`,inode64,logbufs=8,logbsize=256k`	

Note the following:

- SGI recommends that you do not mount any filesystems with `dmi` other than those displayed above.
- Specify the actual filesystem mount point in `mtpt=`*mountpoint*.
- Use of the `dirsync` option will ensure the integrity and consistency of *STORE_DIRECTORY* with the DMF daemon database in the event of a system crash and thereby avoid potential data corruption.
- The `logbufs=8,logbsize=256k` mount options tune the XFS log. These settings are especially important for performance on *MOVE_FS*, which will have heavy metadata activity from the quantity of small files that pass through it.

For more information, see the `mount`(8) and `fstab`(5) man pages.

Migrate at Least Two Copies of a File

When you migrate a file in a DMF configuration, SGI strongly recommends that you make at least two secondary-storage copies of it on separate media to prevent file data loss in the event that a migrated copy is lost.

Note the following:

- Storing multiple copies on the cloud via the `CLOUD_REDUNDANCY` setting for a given cloud MSP does not ensure sufficient data protection in case of file loss. SGI highly recommends that you store a second copy of data on another cloud MSP, on tape, or on disk.
- Because the fast-mount cache configuration requires at least two copies (one to the cache and one to a secondary-storage target), SGI therefore recommends that you migrate at least three copies for this configuration (one to the cache and two to secondary-storage targets). See "Use Fast-Mount Cache Appropriately" on page 112.

Safely Make Changes to the DMF Configuration

This section discusses the following:

- "Make and Mount the Required Filesystems First" on page 101
- "Use Sample DMF Configuration Files" on page 101
- "Back Up the DMF Configuration File" on page 102

- "Stop DMF Before Making Changes" on page 102
- "Always Validate Your Changes" on page 103

Make and Mount the Required Filesystems First

You should make and mount the filesystems required for the DMF administrative directories before making configuration changes. If you try to apply configuration changes without having the filesystems referred to in the configuration file in place, you will get errors. See "Configure Filesystems and Directories Appropriately for DMF" on page 94

Use Sample DMF Configuration Files

DMF is shipped with sample configuration files in the following directory where *DMFVERSION* is a variable that represents the DMF release version:

`/usr/share/doc/`*DMFVERSION*`/info/sample`

The sample files use a variety of MSPs and LSs for different purposes:

- `dmf.conf.cloud` (cloud MSP)
- `dmf.conf.copan_maid` (COPAN massive array of idle disks)
- `dmf.conf.copan_vtl` (SGI 400 virtual tape library)[1]
- `dmf.conf.disk` (disk MSP)
- `dmf.conf.dcm` (disk cache manager MSP)
- `dmf.conf.fmc` (fast-mount cache, such as for COPAN MAID in conjunction with a physical tape library)
- `dmf.conf.ftp` (FTP MSP)
- `dmf.conf.jbfs` (JBFS using an LS)
- `dmf.conf.ls` (LS)
- `dmf.conf.parallel` (Parallel Data-Mover Option)

1 For historic reasons, the SGI 400 VTL is sometimes referred to as *COPAN* in literals and the graphical user interfaces.

You can edit these files via a file editor such as `vi`(1) or DMF Manager (see "Setting Up a New DMF Configuration File" on page 212). You should always validate your changes; see "Always Validate Your Changes" on page 103.

You can use the information in Chapter 7, "DMF Configuration File" on page 271, and in the `dmf.conf`(5) man page to customize your configuration.

Back Up the DMF Configuration File

After you have initially successfully configured DMF, make a backup copy of the DMF configuration file (`/etc/dmf/dmf.conf`) so that you can return to it in case of failure. If you are using DMF Manager, it will automatically make a time-stamped backup for you.

If you have an existing configuration, you should ensure that a good backup copy of the DMF configuration file exists before making any configuration changes.

Stop DMF Before Making Changes

It is safest to make changes to the DMF configuration while DMF is stopped. (For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage*.) If you choose to make changes while DMF is running, be very cautious.

Warning: Never change pathnames or server names in `base` object parameters or add, delete, or change the order of `migrategroup` stanzas while DMF is running; making changes of this type can result in data corruption or data loss.

Do not change the following parameters while DMF is running:

```
ADMDIR_IN_ROOTFS
CACHE_DIR
COPAN_VSNS
DRIVE_GROUPS
EXPORT_METRICS
GROUP_MEMBERS
LICENSE_FILE
LS_NAMES
MSP_NAMES
MULTIPLIER
OV_KEY_FILE
```

```
OV_SERVER
ROTATION_STRATEGY
SERVER_NAME
SERVICES_PORT
SPOOL_DIR
TMP_DIR
VOLUME_GROUPS
```

Before making changes to any parameter, see the information about it in Chapter 7, "DMF Configuration File" on page 271.

Always Validate Your Changes

SGI recommends that you always verify any configuration changes you make:

- When using DMF Manager to make changes, select the following to verify the changes:

  Overview
  > Configuration ...
  > Validate Current Configuration

- When using a file editing tool such as `vi` to directly edit the DMF configuration file, you should do the following:

  1. Back up the current configuration file. For example:

     # **cp /etc/dmf/dmf.conf /etc/dmf/dmf.conf.$(date +%Y%m%d)**

  2. Copy the current configuration to a test location. For example:

     # **cp /etc/dmf/dmf.conf /etc/dmf/dmf.conf.new**

  3. Edit `/etc/dmf/dmf.conf.new`.

  4. Run the `dmcheck`(8) command on the test file:

     # **dmcheck -i /etc/dmf/dmf.conf.new**

     Fix errors and rerun `dmcheck` as needed until it passes without error.

  5. Move the test file to the standard location, overwriting the current file:

     # **mv /etc/dmf/dmf.conf.new /etc/dmf/dmf.conf**

Use Inode-Resident Extended Attributes and 256–byte Inodes

SGI recommends that you configure your filesystems so that the extended attribute used by DMF is always inode-resident and that you use 256–byte inodes and the default `attr2` (`-i attr=2` option to `mkfs.xfs`) when possible. See "Inode Size Configuration" on page 171.

Limit Path Segment Extension Records

You should configure your database record length to minimize the number of records that require a path segment extension record. See "Daemon Database Record Length" on page 173.

Do Not Change Script Names

Do not change the pathnames or script names of the DMF administrative tasks. For more information, see "Automated Maintenance Tasks" on page 175.

Configure DMF Appropriately with CXFS™

DMF must make all of its DMAPI interface calls through the CXFS active metadata server. The CXFS client nodes do not provide a DMAPI interface to CXFS mounted filesystems. A CXFS client routes all of its communication to DMF through the metadata server. This generally requires that DMF run on the CXFS metadata server. If DMF is managing a CXFS filesystem, DMF will ensure that the filesystem's CXFS metadata server is the DMF server and will use metadata server relocation if necessary to achieve that configuration.

Note: DMF data-mover processes must run only on the DMF server node and any parallel data-mover nodes. Do not run data-mover processes on CXFS standby metadata server nodes.

To use DMF with CXFS, do the following:

- For server-capable administration nodes, install the `sgi-dmapi` and `sgi-xfsprogs` packages from the ISSP release. These are part of the software for the DMF server and **DMF parallel data mover**. The DMF software will automatically enable DMAPI, which is required to use the `dmi` mount option.

  For CXFS client-only nodes, no additional software is required.

- When using the Parallel Data-Mover Option, install the software for the DMF parallel data mover, which includes the required underlying CXFS client-only software. (From the CXFS cluster point of view, the DMF parallel data-mover node is a CXFS client-only node but one that is dedicated to DMF data mover activities.) For more information, see:

  – "Parallel DMF Overview" on page 33

  – "Parallel DMF Configuration Procedure" on page 485

- Use the `dmi` option when mounting a filesystem to be managed.

- Start DMF on the CXFS active metadata server for each filesystem to be managed.

See also "SAN Switch Zoning or Separate SAN Fabric Requirement" on page 49.

For more information about CXFS, see:

- *CXFS 7 Administrator Guide for SGI InfiniteStorage*
- *CXFS 7 Client-Only Guide for SGI InfiniteStorage*

Improve Drive Performance with an Appropriate VG Zone Size

When using an LS, it is critical that the zone size you specify for the VG (the `ZONE_SIZE` parameter) is appropriate for the media speed and average data compression rate at your site. A value that is too small can cause poor write performance because a volume mark is written at the end of each zone; a value that is too large can reduce parallelism when migrating files.

The optimal zone size depends upon several site-specific factors. Answering the following questions will help you determine the correct zone size for your site:

- How long does it take the drive to flush data to media?

Note: Different drive types have different bandwidths, and the same drive type can have different bandwidths with different cartridge types.

- How fast can the drive write data?
- What is the average data compression rate? If your data compresses well, the zone size should be larger; if the data does not compress well, the zone size should be smaller.

A good zone size is one where the time spent flushing data to media is not a significant amount of the total I/O time. For increased write performance, choose a zone size such that the average time to write a volume mark for the drive type is a small percentage (such as 5%) of the time to write a zone at the drive's native rate.

For example, suppose the following:

- The drive requires 2 seconds to flush the data to tape
- The drive writes data at 120 MB/s
- The average compression rate is 2 to 1

In order to waste no more than 5% of the full bandwidth of the drive flushing data to media, the `ZONE_SIZE` value in this case must be large enough to hold 40 seconds (2 seconds / 0.05) worth of data in each zone. Because the drive writes at about 120 MB/s, then 40*120=4800 MB of data that can be written in 40 seconds. Not considering compression, a good preliminary `ZONE_SIZE` value is therefore `5g` (5 GB).

Because the example site has a compression rate of 2 to 1, the preliminary `ZONE_SIZE` value should be multiplied by 2; the resulting `ZONE_SIZE` value should be `10g` (10 GB), which is how much data will get written in 40 seconds while still keeping the flush waste within 5% of the total bandwidth.

Note: The zone size influences the required cache space. The value for the `CACHE_SPACE` parameter should be at least twice the value used for `ZONE_SIZE`. Increasing the `ZONE_SIZE` value without also increasing `CACHE_SPACE` could cause volume merging to become inefficient. Volume merges could have problems if the `ZONE_SIZE` value is larger than the `CACHE_SPACE` value. For more information about `CACHE_SPACE`, see "`libraryserver` Object Parameters" on page 393.

For more information about zone size, see the following:

- ZONE_SIZE parameter in "volumegroup Object" on page 410
- "Media Concepts" on page 534
- Appendix H, "Case Study: Impact of Zone Size on Tape Performance" on page 781

Appropriately Force Small Files to Migrate

If you want to quickly migrate small files, you should set TIMEOUT_FLUSH to a value that is less than the default of 2 hours but greater than 15 minutes. See "volumegroup Object Parameters" on page 411.

Caution: Do not set change TIMEOUT_FLUSH to fewer than 15 minutes or set ZONE_SIZE to an artificially low value for the purpose of migrating files more quickly; either of these actions may have a negative impact on performance.

Add HBA Drivers to the initrd Image

The ts tape drive reads HBA information from sysfs just after being loaded in order to discover controller information. To ensure that this information is available when ts loads, SGI recommends that you add the HBA drivers to the initrd image so that they load early in the boot process. Do the following:

1. Add the HBA driver to the INITRD_MODULES line in the /etc/sysconfig/kernel file. For example, to add the driver QLogic QLA2200, you would include qla2xxx in the INITRD_MODULES line.
2. Create the initial RAM disk image so that it contains your modification:

   ```
   # mkinitrd
   ```

3. Reboot the DMF server.

Set RECALL_NOTIFICATION_RATE to 0 if CXFS Range Tokens are Disabled

In a CXFS environment if CXFS range tokens are disabled, you should specify a value of 0 for the DMF configuration parameter RECALL_NOTIFICATION_RATE in order to avoid token thrashing, which can result in poor I/O transfer rates.

Using `0` can slightly improve recall performance in cases where users do not need to access files while they are coming online. (In this case, `dmatrc` does not have to stop and do an `fsync` every 30 seconds during the recall.) The optimum setting of `RECALL_NOTIFICATION_RATE` is dependent on many factors and must be determined by trial and error. See "`dmdaemon` Object" on page 289.

Set the `xinetd tcpmux instances` Parameter Appropriately

You must use a sufficient setting for the `tcpmux instances` parameter in either the `/etc/xinetd.conf` file or the `/etc/xinetd.d/tcpmux` file.

Each remote DMF client command will consume one instance of a `tcpmux` service while it is active. For that reason, SGI recommends that you add the `instances` parameter to `/etc/xinetd.d/tcpmux` rather than increasing the `instances` parameter in `/etc/xinetd.conf`.

Determining the correct setting of this parameter depends on what the maximum number of simultaneous remote DMF user commands might be combined with any other `xinetd tcpmux` services that will be used. See the `xinetd`(8) man page for more information on setting the parameter.

Additionally, it is important that the `tcpmux` service is not disabled. If the following configuration line exists in `/etc/xinetd.d/tcpmux`, remove it:

```
disable = yes
```

Avoid Unintentional File Recall by Filesystem Browsers

Graphical user interface (GUI) filesystem browsers (such as Windows Explorer, GNOME™ Nautilus / File Manager) can unintentionally cause files to be recalled because they read the first few blocks of the file in order to show the correct icon in the view screen:

- Windows Explorer: if you follow the directions in "Modify Settings If Providing File Access via Samba" on page 127, you can avoid this problem for Windows Explorer.
- Nautilus and other filesystem browsers: these filesystem browsers may have settings to prevent them from reading the file for thumbnail icons, but testing is still required because the browser may still read the file for other reasons. Also, file browser behavior may change in future releases, so you must retest after upgrading. You should do one of the following for these filesystem browsers:

– Do not use GUI filesystem browsers on a managed filesystem.

– Set the DMF policy to keep the number of kilobytes permanently on disk required by your filesystem browser, to allow the reading activity to happen without recalling files. Do the following:

1. Determine how many kilobytes are read by your filesystem browser.
2. Verify that the partial-files feature is enabled (see `PARTIAL_STATE_FILES` in "`dmdaemon` Object" on page 289).
3. Use the `ranges` clause to keep the required number of bytes of each file online. See:
   - "`ranges` Clause" on page 383
   - "Automated Space-Management Example" on page 386, and "Automated Space-Management Using Ranges Example" on page 387
4. Repeat the above steps as needed after upgrading the filesystem browser.

See also:

- "Partial-State Files" on page 5
- Appendix G, "Considerations for Partial-State Files" on page 779

Configure Appropriately for SGI 400 VTL or COPAN MAID Shelves

You can use SGI 400 VTL shelves COPAN MAID shelves either as secondary storage or a cache. For initial configuration, see:

- *COPAN MAID for DMF Quick Start Guide*
- *SGI 400 VTL for DMF Quick Start Guide*

To use DMF with SGI 400 VTL shelves or COPAN shelves, do the following:

- Within reason, create smaller volumes, so that hard-deletes will free-up volumes without requiring merges. In general, a larger number of smaller-sized volumes will result in fewer partially-full volumes in the DMF database (and therefore more room for new data). This can potentially provide faster recalls of migrated data because there are more volumes available for reading and writing (you cannot simultaneously write to and read from the same volume). However, you do not want to use volumes that are unreasonably small, as that might cause

excessive mounts and unmounts. For size recommendations, see the *Quick Start* for your system.

- Set the volume size, so that you can use the `dmcapacity`(8) command or its display in DMF Manager to accurately estimate the remaining capacity of the volumes on the COPAN shelves. See "Set Volume Size If You Want to Use Capacity Features" on page 129.
- Use the sample DMF configuration files:
  - For secondary storage, use `dmf.conf.copan_maid` or `dmf.conf.copan_vtl`.
  - For cache, use `dmf.conf.fmc`. See "Use Fast-Mount Cache Appropriately" on page 112.

  Each sample file does the following:
  - Uses one OpenVault library control program (LCP) per shelf
  - Uses a shelf with a single DG and a single VG.
  - Uses one or more migrate groups to combine multiple COPAN shelves into a single destination for a migration request
- Set the following parameters, which apply particularly to COPAN shelves, appropriately according to the information in Chapter 7, "DMF Configuration File" on page 271:

  ```
  COPAN_VSNS
  MAX_PUT_CHILDREN
  RESERVED_VOLUMES
  ZONE_SIZE
  ```

Note: For COPAN shelves, a larger number for `MAX_PUT_CHILDREN` may provide more total write bandwidth, but the bandwidth increases will diminish rapidly with additional children and all of the children will write more slowly. For COPAN MAID, use a `MAX_PUT_CHILDREN` value in the range `2-6`; for SGI 400 VTL, use a `MAX_PUT_CHILDREN` value in the range `2-4`.

If creating backups via `xfsdump` to disk for COPAN MAID, also set the following parameters appropriately:

```
COMPRESSION_TYPE
DUMP_COMPRESS
DUMP_CONCURRENCY
DUMP_DESTINATION
DUMP_MIRRORS
```

- For COPAN MAID, use one VG per shelf.

Use Migrate Groups Appropriately

If you use migrate groups (MGs), do the following:

- Do not specify overlapping MSPs, VGs, or MGs on the same MSP/VG selection parameter. You must ensure that the statement expands to a set of non-overlapping MSPs and VGs when all of the MG members are considered. See:
  - "MSP/VG Selection Parameters for a Managed Filesystem" on page 374
  - "VG Selection Parameters for a DCM MSP `STORE_DIRECTORY`" on page 380
- Never add, delete, or change the order or contents of `migrategroup` stanzas while DMF is running.
- If you want to use a cloud/DCM/FTP MSP as a group member of an MG with a sequential rotation strategy, it should be the last group member listed (because these MSPs are never marked as full by DMF). See "`migrategroup` Object" on page 426.
- Do not include an MSP or VG that uses the `IMPORT_ONLY` parameter (meaning that the MSP/VG is used only for recalls) in a `migrategroup` stanza. The `dmcheck` command will flag this situation as an error.
- If you specify a `ROTATION_STRATEGY` of `SEQUENTIAL`, all `GROUP_MEMBERS` except the last should be able to report when they are full:
  - For a disk MSP, you should specify `FULL_THRESHOLD_BYTES`.
  - For a VG, you should specify a non-zero value for `RESERVED_VOLUMES`.
  - Because a cloud/DCM/FTP MSP never reports that it is full, if used it must be the last member in the `GROUP_MEMBER` list.

  For more information, see:

– "Configure Appropriately for SGI 400 VTL or COPAN MAID Shelves" on page 109

– "`volumegroup` Object" on page 410

– "`msp` Object for a Disk MSP" on page 466

Use Fast-Mount Cache Appropriately

Using a cache (such as COPAN MAID) in conjunction with other secondary-storage migration targets (such as VGs in a physical tape library) is appropriate if your site has a high turnover of relevant data and therefore the most recently migrated files are also the most likely to be recalled.

To use fast-mount cache environment, do the following:

- Define a `fastmountcache` object for each logically separate fast-mount cache. By using multiple logical fast-mount caches, you can account for differences in the following characteristics:
  - The percentage of free volumes that must be available (minimum and target values)
  - File retention policies
  - Physical library residency
- Set the `CACHE_MEMBERS` parameter to name one or more `migrategroup` and `volumegroup` objects that constitute the fast-mount cache. The type of object you name will control what DMF considers when determining whether the free-volume threshold has been reached and the number of volumes to therefore be freed (see `FREE_VOLUME_MINIMUM` and `FREE_VOLUME_TARGET` below):
  - If you name a `migrategroup` object, DMF will consider the total number of volumes that constitute that MG
  - If you name a `volumegroup` object, DMF will consider only the number of volumes that constitute that VG
- Set the following `volumegroup` object parameters:
  - Set `RESERVED_VOLUMES`:
    - `0` (the default) for a VG that is an independent member of a fast-mount cache (that is, the VG is listed in `CACHE_MEMBERS`)

    - `1` for every VG that is part of an MG that is a member of a fast-mount cache (that is, the MG is listed in `CACHE_MEMBERS`)

  – Set `MERGE_THRESHOLD` to `0` for any VG that is part of a fast-mount cache (whether it is the MG or the VG that is listed in `CACHE_MEMBERS`)

  – Do not assign an `ALLOCATION_GROUP` parameter to any VG that is part of a fast-mount cache (whether it is the MG or the VG that is listed in `CACHE_MEMBERS`)

- Ensure that the fast-mount cache is the first target chosen. See "Ensure that the Cache Copy is Recalled First" on page 116.

- Define two other VGs/MSPs (such as on physical tape) as secondary–storage locations into which file data is copied at the time of initial migration, along with the fast-mount cache location. These VGs and MSPs must not be on a DCM MSP or on another fast-mount cache.

  Note: One other VG/MSP is the minimum requirement, but SGI recommends two so that the recommended two migrated copies will remain after the copy in the fast-mount cache has been deleted.

- Do not schedule merging tasks for the `volumegroup` or `migrategroup` objects that represent the fast-mount cache.

- If two separate fast-mount caches are configured, do not configure any policies that would result in a file being migrated to more than one fast-mount cache.

- Define a `taskgroup` object for the fast-mount cache with a `RUN_TASK` object for the `run_fmc_free.sh` script and the following parameters:

  – Required to free the volume when full:

  ```
  FREE_VOLUME_MINIMUM
  FREE_VOLUME_TARGET
  ```

  Note: Because the volumes can be freed immediately, normally you want to set the above to relatively low values. You must set `FREE_VOLUME_MINIMUM` so that it is less than `FREE_VOLUME_TARGET`.

  - Optional to ensure that recently accessed files are copied to another volume in the fast-mount cache before the original volume is emptied (which can result in lower performance):

    `FILE_RETENTION_DAYS`

  - Optional to minimize the competition for disk space by the *MOVE_FS* scratch filesystem when using `FILE_RETENTION_DAYS`:

    `FMC_MOVEFS`

See:

- "`dmdaemon` Object Parameters" on page 290
- "`taskgroup` Object Parameters" on page 312
- "`taskgroup` Object Example for Fast-Mount Cache Tasks" on page 339
- "`drivegroup` Object Parameters" on page 397
- "`volumegroup` Object" on page 410
- "Example: LS for Fast-Mount Cache" on page 439

Use Cloud Storage Appropriately

This section discusses the following:

- "Migrate a Second Copy Outside of the Cloud" on page 114
- "Use a Firewall" on page 115
- "Whitelist Only the DMF Server in a Private Cloud" on page 115
- "Use a Dedicated S3 Bucket" on page 115
- "Improve Throughput for S3" on page 115

Also see "Minimize Access Times by Using `NAME_FORMAT`" on page 116.

Migrate a Second Copy Outside of the Cloud

When using the cloud MSP, it is not sufficient from a DMF data-protection perspective to have multiple copies stored in a single cloud using the cloud's

redundancy feature; to protect data, you must migrate a second copy to a different cloud, tape, or disk storage. See "Migrate at Least Two Copies of a File" on page 100.

Use a Firewall

When setting up a firewall, you should add a rule that allows the DMF server to initiate a TCP connection to the port listed in the `CLOUD_HOST` configuration parameter; see "`msp` Object Parameters for a Cloud MSP" on page 449. The default ports are `81` for Scality and `80` for Simple Storage Service (S3).

Note: You should also be aware of the security issues listed in "Cloud MSP Limitations and Considerations" on page 572.

Whitelist Only the DMF Server in a Private Cloud

When setting up a private cloud in a production environment, you should use `iptables` rules on the cloud server that deny access by default but whitelist the specific IP addresses that are authorized to use the service.

Use a Dedicated S3 Bucket

For simplicity when using the S3 cloud, use a bucket that is dedicated to the data migrated by DMF. However, if other applications will use the same bucket, you can use the `CLOUD_BASE_PATH` parameter to provide a prefix to the keys that will help you identify them as data migrated by DMF. See "`msp` Object Parameters for a Cloud MSP" on page 449.

Improve Throughput for S3

For an S3 cloud MSP, see the following for more information about improving throughput:

http://aws.amazon.com/articles/Amazon-S3/1904

Also see "`NAME_FORMAT` Details" on page 446 for information about minimizing access times.

Minimize Access Times by Using NAME_FORMAT

The NAME_FORMAT configuration parameter for MSPs consists of substitution strings that form a template for filenames used in secondary storage. You can customize the substitution strings to provide a template that is suited to your site. See "NAME_FORMAT Details" on page 446.

Ensure that the Cache Copy is Recalled First

The fast-mount cache and DCM MSP copies must be used for recall before any other copy in order to take advantage of their faster recall characteristics. To achieve this, you must correctly specify the order of parameter values in the DMF configuration file. Do the following:

- List any DCM MSP names first for the LS_NAMES parameter
- List any fast-mount cache LS, DG, and VG names first for the LS_NAMES, DRIVE_GROUPS, and VOLUME_GROUPS parameters, respectively

For more information, see Chapter 7, "DMF Configuration File" on page 271.

Use a Task Group to Run dmmigrate Periodically

Sites whose workflow involves ingesting many files throughout the day in an unpredicatable pattern may find that relying on dmfsfree(8) alone to migrate these files in insufficient. There may be many files that require migration just prior to running the daily xfsdump(8) task, and there may be many new files that require migration.

To avoid these problems, you can use a taskgroup object that calls the run_dmmigrate.sh script to run the dmmigrate(8) command on a regular basis throughout the day to cut down on the amount of work needed prior to an xfsdump run. The object that calls the task group determines the scope of the migration and the location of the associated log messages

- If you reference the task from the dmdaemon object, dmmigrate will be run on all filesystems defined with automated space-management or user space-management enabled. The log messages generated by the script will appear in the dmdlog file.
- If you reference the task from either a filesystem object or a DCM msp object, dmmigrate will migrate data from that object only. Each object can reference the same or different taskgroup objects. The log messages generated by the script will

appear in the autolog file if the task is called from a filesystem object or in the appropriate msplog file if called from a DCM msp object.

Note: If the same taskgroup is referenced by multiple objects, then there will be separate dmmigrate commands running simultaneously for multiple objects. This may result in an unwanted spike in migration requests sent to the daemon.

You can modify the operation of dmmigrate by using the following configuration parameters in the taskgroup object to specify that it will be run with particular dmmigrate command-line options:

DMMIGRATE_MINIMUM_AGE (-m *minutes*)
DMMIGRATE_TRICKLE (-t)
DMMIGRATE_VERBOSE (-v)
DMMIGRATE_WAIT (-w)

Note: When enabled, DMMIGRATE_TRICKLE (ON by default) only limits the number of requests submitted at a time by an individual dmmigrate command. If you define multiple taskgroup objects containing the run_dmmigrate.sh script that are scheduled to run with overlapping times, it is still possible to flood the DMF daemon with migration requests even if DMMIGRATE_TRICKLE is enabled. Therefore, SGI recommends that you to call the taskgroup object containing the run_dmmigrate.sh script from the dmdaemon object in order to migrate files in all managed filesystems with a single command.

For more information about these parameters, see:

- "taskgroup Object Parameters" on page 312
- "taskgroup Object Example for Periodic dmmigrate Tasks" on page 341
- The dmmigrate(8) man page

Restrict the Size of the Alerts and Performance Records Databases

You should configure tasks to automatically purge old records from the alerts and performance databases, based on the age of the records and the size of the databases:

- run_remove_alerts.sh removes records from the alerts database according to the following parameters:

```
ALERT_RETENTION
MAX_ALERTDB_SIZE
REMALERT_PARAMS
```

- `run_remove_perf.sh` removes records from the performance database according to the following parameters:

  ```
  PERF_RETENTION
  MAX_PERFDB_SIZE
  REMPERF_PARAMS
  ```

Note: If you configure a task group to run the above scripts, then you must specify at least one of the retention or database-size parameters. For example, if you specify a task group containing `run_remove_alerts.sh` but you do not include either the `ALERT_RETENTION` or the `MAX_ALERTDB_SIZE` parameter, you will get an error.

The sample configuration files provide task groups with recommended starting values:

- An age of 4 weeks for alert and performance records
- A maximum alerts database size of 100 MB
- A maximum performance database size of 256 MB

However, you should modify these values as necessary for your site. SGI recommends that the alerts and performance databases each be less than 512 MB. For approximate size requirements, see "*HOME_DIR* Size"

See:

- "Overview of the Tasks" on page 304
- "`taskgroup` Object Parameters" on page 312
- "`taskgroup` Object Example for Removing Alerts and Performance Records" on page 341

Prevent Stalled-Recovery Timeout in a Non-HA Environment

If you use CXFS and DMF in a non-HA environment, you must disable the stalled-recovery timeout feature for all potential CXFS metadata servers of managed

filesystems. This will prevent a standby metadata server from experiencing a recovery timeout while waiting for DMF to be manually started.

For example, add the following lines to the `/etc/modprobe.d/sgi-cxfs-xvm.conf` file on all potential CXFS metadata servers for the managed filesystems:

```
# Disable recovery timeout feature to allow for
# manual startup of DMF on the standby MDS during recovery
options sgi-cell cxfs_recovery_timeout_stalled=0
```

See the information about the `cxfs_recovery_timeout_stalled` system tunable parameter in the *CXFS 7 Administrator Guide for SGI InfiniteStorage.*

Use Appropriate Tape Barcodes

A tape library must be set up to report 8–character barcodes to OpenVault: the first 6 characters provide a unique volume serial number (VSN) and the final 2 characters indicate the media type (such as `L5`, which indicates LT05 media). Many libraries report 8 characters by default, but some libraries may require modification; for details, see the information about supported libraries and tape drives in the DMF release note.

Use `dmarchive` to Copy Archive File Data to Secondary Storage

You can use the *direct archiving* feature to manually copy file data between a POSIX filesystem (such as a Lustre™ filesystem) directly to DMF secondary storage by configuring the POSIX filesystem for archive use in the DMF configuration file and using the `dmarchive`(1) command. This POSIX filesystem cannot be DMAPI-enabled (that is, it cannot be mounted with the `dmi` mount option) and is known as an *archive filesystem*. When using this feature, DMF software copies the file data to DMF secondary storage while placing the metadata in a filesystem that is managed by DMF, as shown in Figure 3-2.

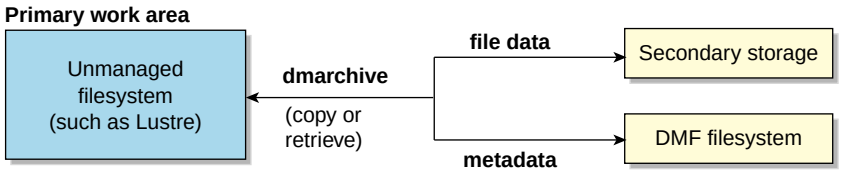

Figure 3-2 Archiving Files from an Archive Filesystem to Secondary Storage

Figure 3-3 shows that the Lustre server is serving the `/lustrefs/work` filesystem, which is mounted on both the DMF server and the DMF client, allowing you to run the `dmarchive` command. The DMF server is managing the `/dmf` filesystem, which is NFS-mounted at `/mnt/dmfusr1` on the DMF client.

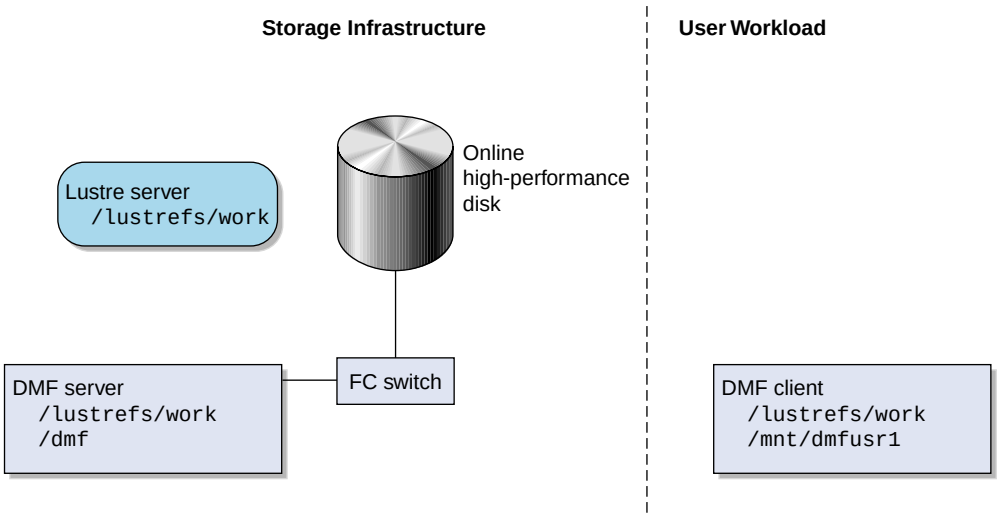

Figure 3-3 DMF Direct Archiving

Without the `dmarchive` command, you would have to first manually copy the file to a managed filesystem and then manually migrate the files. For example:

```
dmfclient% cp -a /lustrefs/work /mnt/dmfusr1
dmfclient% dmput /mnt/dmfusr1/work/*
```

However, using `dmarchive` on a DMF client, you can achieve the same results with a single command:

```
dmfclient% dmarchive -a /lustrefs/work /mnt/dmfusr1
```

Using `dmarchive`, the file data will be copied directly to DMF secondary storage and the file metadata will be copied to the specified managed filesystem (`/dmf`) . The `dmarchive` command recursively copies the entire directory structure (similar to `cp -a`), so the metadata will reside in `/dmf/work`.

On retrieval, the data is copied directly from DMF secondary storage to the archive filesystem. The `dmarchive` method is therefore more efficient because it requires less time, bandwidth, and filesystem capacity.

Use an Appropriate Filesystem for a Disk MSP

If you use a disk MSP, the filesystem should be local for best performance and reliability.

A remote filesystem mounted through NFS or a similar file-sharing protocol is appropriate when testing or when transitioning to a DMF environment.

Use Corresponding Drive-Group Names in OpenVault and DMF

OpenVault and DMF each have a group of interchangeable devices known as a *drive group*. Despite their use of the same terminology, a DMF drive group is different from an OpenVault drive group, and need not use the same name. However, to avoid confusion, SGI recommends that you use corresponding names for the DMF drive group and the OpenVault drive group whenever possible.

Use a Private Network Interface in a Parallel Environment

To prevent the possibility of external attack when using the Parallel Data-Mover Option, do the following:

- Set the `INTERFACE` and `MERGE_INTERFACE` parameters to a private network. See "`node` Object Parameters" on page 294.
- Do not use port scanners on the private network.

Modify Partial-State Capability with Care

The partial-state file capability is turned on by default and an appropriate default maximum number of regions is calculated at filesystem mount time.

You can use the MAX_MANAGED_REGIONS parameter to configure the maximum number of file regions on a per-filesystem basis, but you should use this parameter cautiously. If set capriciously, filesystem scan times can increase greatly. For details about using MAX_MANAGED_REGIONS, see "filesystem Object" on page 350.

To turn off the partial-state file feature, set the PARTIAL_STATE_FILES daemon configuration parameter to off. See "dmdaemon Object Parameters" on page 290.

For additional details, see Appendix G, "Considerations for Partial-State Files" on page 779.

Use the Correct Child-Process Constraints for a DCM MSP

If your environment uses a DCM MSP, you should ensure that you have appropriate settings for the following parameters in the DMF configuration file:

- CHILD_MAXIMUM
- GUARANTEED_DELETES
- GUARANTEED_GETS

SGI recommends that you use a larger value for CHILD_MAXIMUM than the default with a DCM MSP in order to maximize throughput to the DCM *STORE_DIRECTORY*. The value for CHILD_MAXIMUM must be greater than the sum of the values for GUARANTEED_DELETES and GUARANTEED_GETS.

See "msp Object Parameters for a DCM MSP" on page 460.

Explicitly Specify the Server on OpenVault Commands

To ensure that OpenVault communicates with the intended server, explicitly specify the -S *server* option on OpenVault commands such as ov_admin(8), ov_dcp(8), ov_stat(8).

Best Practices for Administration

This section discusses the following:

- "Stop and Start DMF Using the `service` Command" on page 124
- "Use a Time Synchronization Application" on page 124
- "Monitor DMF Daily" on page 125
- "Run Certain Commands Only on a Copy of the DMF Databases" on page 125
- "Be Aware of Differences in an HA Environment" on page 125
- "Start Site-Specific Configuration Parameters and Stanzas with "`LOCAL_`"" on page 125
- "Use TMF Tracing" on page 126
- "Run `dmcollect` If You Suspect a Problem" on page 126
- "Modify Settings If Providing File Access via Samba" on page 127
- "Export from the Active CXFS Metadata Server" on page 128
- "Disable Journaling When Loading an Empty Database" on page 128
- "Use Sufficient Network Bandwidth for Socket Merges" on page 128
- "Temporarily Disable Components Before Maintenance" on page 128
- "Gracefully Stop the COPAN Disks and JBFS Disks" on page 129
- "Reload STK ACSLS Cartridges Properly" on page 129
- "Disable Zone Reclaim to Avoid System Stalls" on page 129
- "Set Volume Size If You Want to Use Capacity Features" on page 129
- "Monitor the Size of the PCP Metrics Archive" on page 130
- "Be Aware that API Commands Change Without Notice" on page 130
- "Be Aware of Memory-Mapping Issues" on page 131
- "Use a Task to Perform Hard-Deletes Periodically" on page 131
- "Enable the Enhanced-NFS RPC Corruption Workaround Parameter if Needed" on page 131

- "Use the Appropriate Tool to Load Volumes to an Existing Environment" on page 132
- "Configure Fibre Channel Switches and Zones Appropriately" on page 132
- "Use dmmove(8) Safely" on page 136
- "RHEL: Disable SELinux or Increase the Inode Size" on page 136
- "Apply Appropriate PostgreSQL Updates" on page 136

Stop and Start DMF Using the service Command

To stop or start DMF, you should use the service(8) command. See "Starting and Stopping the DMF Environment" on page 181.

Use a Time Synchronization Application

You must ensure that all nodes that can manage DMF and OpenVault operations have consistent time and time-zone settings. This includes the following:

- DMF server
- Passive DMF server (in an HA configuration)
- Parallel data-mover nodes (if using the Parallel Data-Mover Option)
- OpenVault server (if different from the DMF server)

SGI recommends that you use a time synchronization application on these nodes and that you force synchronization at every boot. For example, if you use Network Time Protocol (NTP), you should set the following in /etc/sysconfig/ntp:

```
NTPD_FORCE_SYNC_ON_STARTUP="yes"
```

Note: If you are using an S3 cloud MSP, the DMF server clock and the S3 clock must be within 15 minutes of each other, otherwise authentication errors will result due to security requirements.

Monitor DMF Daily

You should monitor DMF on a daily basis to ensure that it is operating properly and that you find any problems in time to retrieve data.

DMF provides a number of automated tasks that you can configure to generate reports about errors, activity, and status. Additionally, some serious error conditions generate email messages. Examining this information on a timely basis is important to ensure that DMF is operating properly and to diagnose potential problems.

Run Certain Commands Only on a Copy of the DMF Databases

You should run the following commands **only on a copy** of the DMF databases:

- `dmdbcheck`(8)
- `dmdump`(8)

If you run these commands on an active database (that is, on a database located in the *HOME_DIR* directory while DMF is running), the results of the commands will be unreliable because DMF may be actively changing the data while the command is running.

Be Aware of Differences in an HA Environment

If you run DMF in a high-availability (HA) cluster, some configuration requirements and administrative procedures differ from the information in this guide. For example, in an HA environment you must first remove HA control of the resource group before stopping DMF. For more information, see *High Availability Guide for SGI InfiniteStorage*.

Start Site-Specific Configuration Parameters and Stanzas with "LOCAL_"

If you choose to add site-specific parameters or object stanzas to the DMF configuration file, you should begin the parameter name or stanza name with "LOCAL_" (such as LOCAL_MYPARAM) so that the names will not cause conflict with future SGI DMF parameters and stanzas.

Use TMF Tracing

Each TMF process writes debugging information to its own trace file, located in the directory specified by the `trace_directory` parameter in the TMF configuration file `/etc/tmf/tmf.config`. If you use TMF, you should leave TMF tracing on so that this debugging information is available if problems occur.

The trace files are circular, meaning they only contain the most recent activity from a TMF process. To change the amount of history available in a trace file, modify the `trace_file_size` configuration parameter.

When TMF is restarted, any trace files from the previous instance of TMF are moved to the directory specified in `trace_save_directory`.

For more information, see *TMF 6 Administrator Guide for SGI InfiniteStorage.*

Run `dmcollect` If You Suspect a Problem

As soon as you suspect a problem with DMF, run the `dmcollect`(8) command to gather the relevant information about your DMF environment that will help you and SGI analyze the problem.

Note: Take care to enter the correct number of **previous** days from which to gather information, so that logs containing the first signs of trouble are included in the collection.

For example, the will collect data for today only (0 previous days):

```
server# dmcollect -b 0
```

If the problem started the previous day, you would want to collect data from that day as well (1 previous day):

```
server# dmcollect -b 1
```

Specify a larger number of prior days to cover a longer time period, as required.

Also see Chapter 19, "Troubleshooting" on page 641.

Modify Settings If Providing File Access via Samba

You can avoid unnecessary Windows SMB request timeouts by setting the `SessTimeout`, `ExtendedSessTimeout`, and `ServersWithExtendedSessTimeout` registry parameters on every Windows client system to the values appropriate for a DMF environment. This is especially important for slower mounting/positioning libraries and tape drives. For example:

```
SessTimeout=300
ExtendedSessTimeout=1000
ServersWithExtendedSessTimeout=value
```

Where *value* specifies all possible Samba server names and IP addresses, separated by a new line, such as in the following formats:

SambaServerName
SambaServerName.domainname.com
SambaServerIPaddress1
SambaServerIPaddress2

For details, see the following:

http://blogs.msdn.com/b/openspecification/archive/2013/03/19/cifs-and-smb-timeouts-in-windows.aspx
http://blogs.msdn.com/b/openspecification/archive/2013/03/27/smb-2-x-and-smb-3-0-timeouts-in-windows.aspx
http://technet.microsoft.com/en-au/library/cc938292.aspx

The Windows Explorer desktop can show which files in an SMB/CIFS network share are in a fully or partially offline state. If so enabled, Windows Explorer overlays a small black clock on top of a migrated file's normal icon; the black clock symbol indicates that there may be a delay in accessing the contents of the file. (This feature is disabled by default.)

To enable this feature, do the following:

1. Install the `sgi-samba` RPMs from ISSP.

   Note: This feature is not available in community Samba.

2. Add the following line to the Samba configuration file `/etc/samba/smb.conf` on the DMF server:

   ```
   dmapi support = Yes
   ```

3. Restart the `smb` daemon on the DMF server:

Caution: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage.*

```
server# service smb restart
```

For more information, see the `smb.conf`(5) man page.

Export from the Active CXFS Metadata Server

When exporting a managed filesystem using enhanced Samba in a CXFS environment, you should perform the CIFS export from the active CXFS metadata server in order to realize the full benefit of enhanced Samba.

Disable Journaling When Loading an Empty Database

If you are loading an empty database, you should disable journaling in order to eliminate unnecessary overhead. To do this, use the `-j` option to the `dmdadm`(8) and `dmcatadm`(8) commands. For example:

```
# dmdadm -j -u -c "load /dmf/scratch/daemon.txt"
# dmcatadm -j -m ls -u -c "load /dmf/scratch/ls_cat_txt" > /dmf/tmp/load.ls.db.out 2>&1
```

Use Sufficient Network Bandwidth for Socket Merges

If you perform a merge using a socket, you must ensure that the network has sufficient bandwidth. For more information, contact SGI technical support.

Temporarily Disable Components Before Maintenance

Before you perform maintenance on tape drives, a tape library, a COPAN shelf, or a JBFS enclosure, you can perform steps that will allow DMF to quit using the component. You can then verify that the component is currently unused and will no longer accept new work. See "Temporarily Disabling Components" on page 589.

Gracefully Stop the COPAN Disks and JBFS Disks

Before stopping COPAN disk or JBFS disks, you should ensure that DMF is not using any of its virtual tape drives and then stop the OpenVault LCPs associated with the disks. See "Stop the COPAN VTL" on page 595.

Reload STK ACSLS Cartridges Properly

After you load tape cartridges into a StorageTek tape library controlled by Automated Cartridge System Library Software (ACSLS) via the cartridge access port, you must manually cancel all prior ACSLS `enter` requests. This will allow OpenVault to update the DMF database.

Disable Zone Reclaim to Avoid System Stalls

To avoid transient system stalls on DMF servers and all parallel data-mover nodes, you should disable zone reclaim by adding the following line to the `/etc/sysctl.conf` file:

```
vm.zone_reclaim_mode = 0
```

To make this change take effect, enter the following:

```
# sysctl -p
```

For more information about this kernel parameter, refer to the `Documentation/sysct/vm.txt` file in the Linux kernel source.

Note: For large NUMA compute servers, whose typical workload is HPC applications, you should consider whether the benefits of memory locality outweigh the cost of memory reclaim.

Set Volume Size If You Want to Use Capacity Features

If you want to use features such as the `dmcapacity`(8) command and its display via DMF Manager (which are particularly useful features for COPAN MAID and SGI 400 VTL), you should set the size of volumes in the DMF database. Do one of following:

- Use the `-s` *tapesize* option to `dmov_loadtapes`(8) when loading new volumes to set the size in bytes. For example:

```
# dmov_loadtapes -t Ultrium4-800 -l C00 -s 20000000000 dump_tasks
```

- Use the dmvoladm(8) command with the `update` directive to modify the `ts` field to set the size in bytes. For example:

  ```
  # dmvoladm -m vtl_ls -c "update C00X1Z tapesize 20000000000"
  ```

- Specify the cartridge size on the **Add Volumes** dialog in DMF Manager to set the size in bytes. See "Managing Volumes" on page 229.

Monitor the Size of the PCP Metrics Archive

PCP continuously gathers DMF performance metrics for display in DMF Manager. These metrics are stored in `/var/lib/pcp-storage/archives`.

Note: In an HA environment, the PCP metrics archive is stored in the directory specified by the `dmfman_setup_ha` script. For more information, see the section about configuring DMF for HA in *High Availability Guide for SGI InfiniteStorage*.

Each month, DMF performs a data-reduction process on the metrics gathered for the month. This reduces the size of the archives while retaining a consistent amount of information. Although the size of the archive has a bounded maximum, this can still be quite large depending on the configuration of the server and how many clients access it. For example, a server with a large number of filesystems could generate up to 100 Mbytes of archives per day. You should initially allow around 2 GB of space in `/var/lib/pcp-storage/archives` for archive storage and monitor the actual usage for the first few weeks of operation.

Be Aware that API Commands Change Without Notice

DMF uses several undocumented commands as an internal API layer. These commands can change or be removed without notice.

Note: If you require functionality that is not provided by the standard set of documented DMF administrator and user commands, contact SGI Support to suggest a request for enhancement.

Be Aware of Memory-Mapping Issues

On Linux nodes, memory-mapping an offline file in a DMF filesystem may cause other processes such as `ps`(1) to block while DMF is making the file online.

There are also memory-mapping issues with CXFS clients running Red Hat Enterprise Linux (RHEL) or Windows; for more information, see the *CXFS 7 Administrator Guide for SGI InfiniteStorage*.

Use a Task to Perform Hard-Deletes Periodically

To avoid reaching the maximum DMF database size (4 billion records), you should configure the `run_hard_deletes.sh` task to periodically delete database records that have become eligible for hard-deletion. Otherwise, the daemon database can continue to grow in size as obsolete records accumulate.

See:

- "`taskgroup` Object" on page 304
- "Cleaning Up Obsolete Database Entries" on page 586

Enable the Enhanced-NFS RPC Corruption Workaround Parameter if Needed

A Linux NFS-client data corruption bug was fixed in Linux kernel version 2.6.37 with commit ID `f5fc3c50c99a7df2bf908dfe66f112d35178ee07`. If you are running Linux NFS clients with a prior release, you must avoid the problem by using a system-tunable kernel parameter that is available in enhanced NFS. Contact your Linux NFS client OS distribution provider to determine whether or not you must make this change.

Note: Enabling the parameter can have a negative impact on benchmarking and code compiles over NFS. Therefore, the parameter is disabled (set to `0`) by default. You should enable it only if the kernel version on any of your Linux NFS clients contains the corruption bug.

If required, the parameter that you must enable (set to `1`) on the enhanced-NFS server is `nfsd_workaround_nfs3_xdr_readres_corruption`. Do the following:

1. Add the following line to the `/etc/modprobe.conf` file on the enhanded-NFS server:

   ```
   options nfsd nfsd_workaround_nfs3_xdr_readres_corruption=1
   ```

2. Reboot the enhanced-NFS server.

For more information about the bug and the enhanced-NFS `nfsd_workaround_nfs3_xdr_readres_corruption` parameter, see the section about enhanced NFS in the ISSP release note.

Use the Appropriate Tool to Load Volumes to an Existing Environment

Table 3-4 describes the tools you should use to load volumes to an existing DMF and OpenVault environment.

Table 3-4 Tools to Load Volumes to an Existing DMF/OpenVault Environment

Volumes Already in DMF?	Volumes Already in OpenVault?	Tool to Use
No	No	`dmov_loadtapes`
Yes	No	`dmov_makecarts`
No	Yes	`dmvoladm`

Configure Fibre Channel Switches and Zones Appropriately

You must configure Fibre Channel switches appropriately to ensure that tapes will not be inappropriately overwritten, potentially resulting in data loss. This is particularly important when using the Parallel Data-Mover Option. This section discusses the following:

- "Ensure that You Follow the Switch Requirements" on page 133
- "Segregate Tape and Disk HBAs" on page 133
- "Suppress Change Notification for Switch Ports Connected to Nodes" on page 133
- "Use `N-port` Topology for LSI Ports Used with Tape Drives" on page 133

- "Avoid Bottlenecks when Tape Drives and Host Port Speeds Do Not Match" on page 134

Ensure that You Follow the Switch Requirements

See "SAN Switch Zoning or Separate SAN Fabric Requirement" on page 49.

Segregate Tape and Disk HBAs

Never mix tape and disk access via the same HBA port. SGI also recommends that you do not mix tape and disk access on a given multiport card; this recommendation may not apply to some newer multiport cards.

Use separate physical SAN switches or switch zoning to logically separate tape and disk SAN fabrics. See "SAN Switch Zoning or Separate SAN Fabric Requirement" on page 49.

Suppress Change Notification for Switch Ports Connected to Nodes

To prevent unnecessary interruptions, enable the suppression of change notification if the port is connected to a host HBA and disable the suppression for all other ports. See:

- "Suppressing RSCN on Host Port HBAs" on page 177
- "QLogic® Fibre Channel Switch" on page 178

Use `N-port` Topology for LSI Ports Used with Tape Drives

During error recovery, a bus reset will cause the LSI Fibre Channel port to renegotiate its connection with the Fibre Channel switch. This renegotiation can result in the LSI host port acquiring a different port ID. Should this happen, reservation conflicts or errors that result in the tape driving transitioning to `swdn` can occur. To avoid this problem, use `lsiutil` to set the link topology to `N-port` for all LSI Fibre Channel ports used with tape drives, which eliminates the possibility that the host adapter port could acquire a different port ID.

Avoid Bottlenecks when Tape Drives and Host Port Speeds Do Not Match

Note: This section does not apply to STK drives. For those drives, the only control is the size of the tape drive I/O request, which DMF determines. STK 4–Gbit adapters perform at approximately 200 MB/s.

If you have one 4-Gbit host port and are writing data to multiple 2-Gbit tape drives, the aggregate desired bandwidth on the host port is greater than the data rate of the Fibre Channel (FC) adapters on the tape drives. This can cause the switch's frame buffers to fill up, causing the switch to stop accepting data from the 4-Gbit HBA, dropping the effective data rate close to that of a 2-Gbit HBA.

You can correct this situation by changing the maximum burst size (`burst_size`) for the tape drive. The maximum burst size specifies the maximum amount of data that the port can transfer during a single operation. It should be double the switch port buffering (after unit conversions, because maximum burst size is in units of 512 bytes). For example, a Brocade 4100 switch has at least 32 KB of buffering per port, so you would start with a value of `128`.

Note: Determining the optimum value for `burst_size` depends upon many site-specific factors, including HBA speed, switch speed, tape speed, and number of tapes per port; it may take some trial-and-error to set optimally. SGI suggests beginning by using a value of `64` or `128`, which have been shown to improve results without negative impact.

Before changing the maximum burst size, ensure that you have stopped DMF, APD, and the TMF or OpenVault mounting service.

If you have installed the optional `sdparm` RPM from RHEL or SLES, you can use the `sdparm` command to set the burst size:

```
# sdparm -t fcp --set MBS=burstsize /dev/sgNN
```

You can test the effects of changing the burst size by doing the following:

1. Ensure that the services for DMF, APD, and the TMF or OpenVault are stopped. (In an HA environment, see *High Availability Guide for SGI InfiniteStorage*.)
2. Ensure you have two 2–Gbit tape drives on 4–Gbit FC switch with one 4–Gbit host connection.

3. Set the maximum burst size to `0` (no limit) on both drives. For example:

   # **sdparm -t fcp --set MBS=0 /dev/sg0**

4. Load scratch tapes on the drives.

5. Enter the following for each drive separately and then both drives in parallel and monitor performance with SGI Performance Co-Pilot™ (PCP™) or an FC switch tool:

   # **dd if=/dev/zero of=/dev/ts/... bs=256k**

6. Change maximum burst size. For example, to set it to `128`:

   # **sdparm -t fcp --set MBS=128 /dev/sg0**

7. Enter the following for each drive separately and then both drives in parallel and monitor performance with PCP or an FC switch tool:

   # **dd if=/dev/zero of=/dev/ts/... bs=256k**

To determine the current maximum burst size, use the `sginfo -D` command. For example:

```
# sginfo -D /dev/sg0
Disconnect-Reconnect mode page (0x2)
------------------------------------
Buffer full ratio                   0
Buffer empty ratio                  0
Bus Inactivity Limit (SAS: 100us)   0
Disconnect Time Limit               0
Connect Time Limit (SAS: 100us)     0
Maximum Burst Size                  128
EMDP                                0
Fair Arbitration (fcp:faa,fab,fac) 0
DIMM                                0
DTDC                                0
First Burst Size                    0
```

You can also use the `sdparm --get` command. For example:

```
# sdparm -t fcp --get MBS /dev/sg0
```

For more information about `sdparm`, see:

http://freshmeat.net/projects/sdparm/

http://dag.wieers.com/rpm/packages/sdparm/

Use dmmove(8) Safely

Unless you specify the `-d` option, `dmmove`(8) will delete existing copies from MSPs/VG that are not listed in the final set of arguments on the command line. Therefore, when issuing a `dmmove` command, it is a best practice to always explicitly state all of the MSPs/VGs from which a copy should be deleted (which may be `-d none` if no copy should be deleted) and all of the MSPs/VGs into which a copy should be placed/retained. For an example, see "Replacing a Bad DMF Copy with a Good DMF Copy" on page 642. For details about the command, see the `dmmove`(8) man page.

RHEL: Disable SELinux or Increase the Inode Size

By default, Security-Enhanced Linux (SELinux) is enabled on RHEL systems. Because SELinux uses extended attributes, using the default inode size (256) with SELinux enabled causes all DMF attributes to be stored outside the inode, which potentially slows down most DMF operations.

If you are running DMF with RHEL, you should do one of the following for best performance:

- If SELinux is unnecessary, disable it by setting the following in the `/etc/sysconfig/selinux` file:

  ```
  SELINUX=disabled
  ```

- If you require SELinux, increase the inode size for the filesystems managed by DMF to 512 by using the `-i size=512` option to the `mkfs.xfs` command. For more information, see the `mkfs.xfs`(8) man page.

Apply Appropriate PostgreSQL Updates

You should apply all future security updates and bugfixes in the PostgreSQL 9.3.*X* series.

Note: DMF requires the PostgreSQL 9.3.*X* series. Do not manually upgrade to another major version (such as 9.4.*X*).

Best Practices for Backups

This section discusses the following:

- "Back Up Migrated Filesystems and DMF Databases" on page 137
- "Migrate All Files Before Performing a Backup" on page 137
- "Use the Integrated Backup Method" on page 138
- "Use a Remote Location for the Disaster-Recovery Bundle" on page 138
- "Retain Log and Journal Files Between Full Backups" on page 138

Back Up Migrated Filesystems and DMF Databases

When using DMF, you must still perform regular backups to protect inodes, directory structures, and unmigrated files; DMF moves only the data associated with files, not the file inodes or directories. You can configure DMF to automatically run backups of your managed filesystems.

You must also back up the daemon database and the LS database regularly using the `run_copy_databases.sh` task.

See:

- "Administration Tasks" on page 51
- "`taskgroup` Object" on page 304
- Chapter 4, "Backups and DMF" on page 141

Migrate All Files Before Performing a Backup

You can greatly reduce the amount of time it takes to back up filesystems by configuring DMF to migrate all files in the managed filesystems by setting the `DUMP_MIGRATE_FIRST` parameter to `yes`. See "Migrate All Files First" on page 146.

Use the Integrated Backup Method

When using DMF, you must still perform regular backups to protect inodes, directory structures, and unmigrated files; DMF moves only the data associated with files, not the file inodes or directories. The DMF integrated backup method is the preferred backup method. In this method, DMF is configured to migrate file data to the backup media before using `xfsdump` to dump the remaining file metadata and directory structure so that both dump size and the dump time are reduced. For filesystems not managed by DMF, DMF dumps file data, file metadata, and directory structure.

For information about converting an existing DMF configuration to this method, see "Converting to Integrated Backups" on page 157

Use a Remote Location for the Disaster-Recovery Bundle

For data security SGI recommends that you specify a remote location for the disaster-recovery bundle mirror when using the integrated-backup method. See the `DUMP_DR_MIRRORS` configuration parameter in "`taskgroup` Object Parameters" on page 312

Retain Log and Journal Files Between Full Backups

You must retain DMF log and journal files between full backups of the DMF databases. After a full backup, you may remove old journal and log files to prevent the spool directory from filling. You can use the `run_remove_logs.sh` and `run_remove_journals.sh` tasks to schedule automatic removal of the old files after the backup completes. See "`taskgroup` Object" on page 304.

Best Practices for Optional Tasks

This section discusses the following:

- "Balance Data Among Libraries" on page 139
- "Prevent Recalls From Waiting for a Busy Volume" on page 140
- "Use the Immutable Bit if You Want to Prevent File Modification" on page 140

Balance Data Among Libraries

If you want to balance migrated data among libraries, you can use configurations such as the following:

- Suppose you have two libraries and you want to make two copies of the migrated data. You would use a VG for each library (`lib1vg` and `lib2vg`) and the following statement for VG selection:

  ```
  SELECT_VG    lib1vg lib2vg
  ```

  One copy would go to each library.

  See "MSP/VG Selection Parameters for a Managed Filesystem" on page 374.

- Suppose you have four libraries and you want to make two copies of the migrated data. You can use two migrate groups, each with two VGs, and a rotation strategy of `ROUND_ROBIN_BY_BYTES`:

  ```
  mg1
      lib1vg
      lib2vg

  mg2
      lib3vg
      lib4vg
  ```

  You would use the following statement for VG selection:

  ```
  SELECT_VG    mg1 mg2
  ```

  One copy of would go to either `lib1vg` or `lib2vg`, the other copy would go to either `lib3vg` or `lib4vg`.

  For more information, see:

  – "Migrate Groups" on page 42

  – "Use Migrate Groups Appropriately" on page 111

  – "`migrategroup` Object Example with Multiple MGs" on page 431

Prevent Recalls From Waiting for a Busy Volume

If you want to prevent recall requests from waiting for a volume that is busy because it is being written to, you can use the configuration parameters FORWARD_RECALLS and GET_WAIT_TIME in the volumegroup stanza.

For example, to allow DMF to continue writing to the volume for only up to 1 hour (3600 seconds) after receiving a recall request and to direct recall requests to another VG if this VG is writing to the volume, you would include the following parameters:

```
GET_WAIT_TIME           3600
FORWARD_RECALLS         ON
```

For more information about these parameters, see "volumegroup Object" on page 410.

Use the Immutable Bit if You Want to Prevent File Modification

If you want to prevent a file from being changed or deleted even by the root user, use the XFS immutable bit. For more information, see the chattr(8) command.

Backups and DMF

This section discusses the following:

- "DMF Requirement for Backups" on page 141
- "Dump Scripts" on page 141
- "Site-Specific Factors to Consider" on page 144
- "Media-Specific Requirements" on page 144
- "Considerations for Optimizing Backups" on page 145
- "Integrated-Backup Method" on page 148
- "Other Backup Methods" on page 161

DMF Requirement for Backups

The fact that DMF maintains copies of data on another medium does not mean that it is a backup system. The copies made by DMF may become inaccessible if there is a failure and proper backups have not been made.

In addition, although using RAID may protect you against the failure of one disk spindle, data can still be endangered by software problems, human error, or hardware failure.

Therefore, **backups are essential**.

Dump Scripts

DMF provides the following scripts that you can add to a `taskgroup` object:

- "`run_full_dump.sh`" on page 142
- "`run_partial_dump.sh`" on page 143

run_full_dump.sh

The run_full_dump.sh script performs a full backup of configured filesystems at intervals specified by a time expression when configured as part of a dump taskgroup object.

You can invoke this script manually by using the following command line:

```
/usr/lib/dmf/run_full_dump.sh dmdaemon taskgroup [prune]
```

where:

dmdaemon Indicates the name of the dmdaemon object

taskgroup Indicates the name of the taskgroup object

prune Indicates the type of prune action to be performed, one of the following:

prune-files	*(integrated backups only)* Removes orphaned filesystem dumps from the backup filesystem that have no corresponding entry in the xfsdump inventory. For safety reasons, this type of prune will not remove any other files, so there could still be other orphaned backup files in the backup filesystem (such as database, configurations, and inventories); these files will eventually be removed after their age exceeds the DUMP_RETENTION period. Before using this argument, ensure that the orphaned filesystem dump is no longer required to restore a filesystem; see the **Caution** below.
prune-inv	*(integrated backups only)* Removes orphaned inventory entries from the xfsdump inventory. *Orphaned inventory entries* do not have corresponding dump files in the backup filesystem. Before using this argument, ensure that the entries do not correspond to dump files that might still be restored to the backup filesystem; see the **Caution** below.

`prune-only`	Removes expired dumps based on the `DUMP_RETENTION` value

The `prune-files` and `prune-inv` arguments are typically used to resolve consistency errors reported by an integrated backup session.

Caution: The `prune-inv` and `prune-files` arguments do not prune based on the `DUMP_RETENTION` value and hence could cause important filesystem dumps or `xfsdump` inventory entries to be removed even though they might still be required to restore filesystem components.

`run_partial_dump.sh`

The `run_partial_dump.sh` script performs an incremental backup.

Note: If the `DUMP_MAX_FILESPACE` configuration parameter is defined (in any configuration object that applies to the filesystem) but the `UPDATE_CTIME` configuration parameter is undefined or set to `OFF`, then `run_partial_dump.sh` will for safety reasons perform a full (level-0) backup for a managed filesystem. See "`DUMP_MAX_FILESPACE` Requires `UPDATE_CTIME` for Incremental Backups" on page 148.

You can invoke this script manually by using the following command line:

`/usr/lib/dmf/run_partial_dump.sh` *dmdaemon taskgroup*

where:

dmdaemon	Indicates the name of the `dmdaemon` object
taskgroup	Indicates the name of the `taskgroup` object

Site-Specific Factors to Consider

Backup requirements depend upon a number of very site-specific factors, including the following:

- The amount of data that is migrated and the amount of data that is not migrated at the time a backup takes place
- The number of inodes
- The size of the DMF databases (see "*HOME_DIR* Size" on page 97)
- The backup methodology for using full and/or incremental backups
- Amount of space occupied by each dump, which in turn depends on:
  - Size of DMF database
  - Number of inodes
  - Amount of space occupied by regular files
  - Your choice of full versus incremental backups
- The desired amount of time you want to retain dumps

Media-Specific Requirements

This section discusses the following:

- "Backups using Physical Tapes and SGI 400 VTL Tapes" on page 144
- "Backups using COPAN MAID" on page 145

Backups using Physical Tapes and SGI 400 VTL Tapes

The number of physical or virtual backup tapes that will be used depends upon the retention period and the information in "Site-Specific Factors to Consider" on page 144.

Tapes are recycled after the retention period is completed, therefore you must have more backup tapes than are required to fulfill the retention period (at least one extra tape). Assuming that backups are done daily, the minimum number of tapes required is:

Retention_Period_In_Days + 1 = *#_Backup_Tapes*

For example, using a retention period of 4 weeks (28 days):

```
28 + 1 = 29 tapes
```

So long as each day's backup can fit onto one tape, this means that at a minimum 29 backup tapes are required, assuming that backups are performed each day.

Note: You should monitor the backup report daily to verify that there are sufficient tapes available for future backups. If it turns out that a given day requires multiple backup tapes for the set of backups for that day's backup, you must empty previously used backup tapes or add more backup tapes.

Backups using COPAN MAID

The amount of space that your site will require for the backups created by each day's backup depends upon the information discussed in "Site-Specific Factors to Consider" on page 144. This amount is the *Dump_Space_Needed_Per_Day* value.

The approximate formula for the amount of disk space that you must reserve for backups is:

Dump_Space_Needed_Per_Day * (*Retention_Period_In_Days* + 1) = *Reserved_Space*

You can allocate the *Reserved_Space* on a reserved portion of the RAID set that is not managed by DMF). If you prefer, you could allocate space on physical tapes instead. For more information, see *COPAN MAID for DMF Quick Start Guide*.

Considerations for Optimizing Backups

This section discusses the following:

- "Migrate All Files First" on page 146
- "Consistent Backup Completing Times via DUMP_MAX_FILESPACE" on page 147
- "DUMP_MAX_FILESPACE Requires UPDATE_CTIME for Incremental Backups" on page 148

Migrate All Files First

You can greatly reduce the amount of time it takes to back up filesystems by configuring DMF to migrate all files. To do this, set the DUMP_MIGRATE_FIRST parameter to yes, which specifies that the dmmigrate command is run before the dumps are done to ensure that all migratable files in the managed filesystems are migrated. For more information, see the description of the parameter in the following sections:

- "dumpgroup Object Parameters" on page 343 (integrated-backup context only)
- "filesystem Object Parameters" on page 351 (integrated-backup context only)
- "taskgroup Object Parameters" on page 312

Migrating all files before performing a backup has the following benefits:

- The backup image will be smaller because it contains just the metadata information, not the file data itself
- The backup will complete more quickly because:
  - It is reading just the metadata
  - There is less time spent performing random disk seeks to back up the data of unmigrated files

For any files that you want to remain permanently on disk (that is, permanently dual-state), you can assign a negative priority weight to those files. The result is that when the filesystem is filled up, DMF will never free the blocks for these files. The files therefore are always dual-state, ready to be used. When the filesystem is backed up, the backup facility will recognize that they are dual-state and therefore back them up as offline. The net effect is that there is no file data in the backup at all for these files, just their inodes, while keeping the files always available. In the case of millions of small files, this speed-up of the backup process can be dramatic. For example, for a filesystem with a large number of small files (files of up to 64 KB), you could assign the following AGE_WEIGHT value:

```
AGE_WEIGHT      -1      0       when space < 64k
```

Be aware of the following:

- For extremely small files (under a few hundred bytes), the disk space required for DMF database entries may exceed the size of the original file. For extremely large numbers of such files, this issue should be considered.
- The `space` value in a `when` clause, as used above, refers to the space the file occupies on disk, which for sparse files may actually be smaller than the size of the file as shown by `ls -l`. The `space` value will be rounded upward to a multiple of the disk blocksize defined by `mkfs`(8); the default is 4096 bytes. For example, attempting to discriminate between files above or below 1000 bytes based on their `space` value is futile because all non-empty files will have a `space` value that is a multiple of (typically) 4096 bytes.

If you use negative weights with `AGE_WEIGHT` or `SPACE_WEIGHT`, DMF automatic migration will never free the space for these files but a user can still do a `dmput -r` to manually free the space.

However, if you do not want files to migrate for any reason, then you must continue to use the `SELECT_VG` method despite the slower and larger backups.

Consistent Backup Completing Times via `DUMP_MAX_FILESPACE`

If you do not first migrate all files before performing a backup (see "Migrate All Files First" on page 146) but you still want to make your backup completion times consistent, you can use the `DUMP_MAX_FILESPACE` configuration parameter on specific `filesystem` objects to restrict the size of regular files that are included in the dumps. For example, you could set `DUMP_MAX_FILESPACE` to `1K` to prevent all regular files exceeding 1 KB from being backed up; this assumes that although most regular-state files will be skipped in the next backup, they will soon be migrated, and then their inodes only will be included in a later backup. Using this method will also require setting `UPDATE_CTIME` if using incremental backups; see "`DUMP_MAX_FILESPACE` Requires `UPDATE_CTIME` for Incremental Backups" on page 148

Caution: This parameter is also available for the `dumpgroup` and `taskgroup` objects, but it can be dangerous because files are omitted silently and the limit applies to all of the filesystems specified by an object (which by default is all filesystems for the `taskgroup` object). . Therefore SGI recommends that you set this limit only for specific `filesystem` objects.

DUMP_MAX_FILESPACE Requires UPDATE_CTIME for Incremental Backups

If you set the DUMP_MAX_FILESPACE configuration parameter and you want to perform incremental backups of a managed filesystem, then you must also set the UPDATE_CTIME configuration parameter to ON. This is required to avoid the situation in which a file is never backed up, due to the default way in which timestamps are changed on files.

Note: If you set DUMP_MAX_FILESPACE in any configuration object that applies to the filesystem without setting UPDATE_CTIME to ON, DMF will always perform full (level-0) backups.

The run_partial_dump.sh script for safety reasons performs a full (level-0) backup for a managed filesystem when both of the following conditions are true:

- The DUMP_MAX_FILESPACE configuration parameter is defined in the taskgroup object for tape/disk backups or in the dumpgroup, filesystem, or taskgroup object for integrated backups
- The UPDATE_CTIME configuration parameter is undefined or set to OFF for the filesystem

---

See:

- "dumpgroup Object Parameters" on page 343
- "filesystem Object Parameters" on page 351
- "taskgroup Object Parameters" on page 312
- "filesystem Object Example for Incremental Backups that Restrict File Size" on page 362

Integrated-Backup Method

This section discusses the following:

- "Overview of Integrated Backup" on page 149
- "Contents of the Integrated Backup" on page 149
- "Size of the Backup Filesystems" on page 153
- "DR Bundle Locations" on page 154

- "Configuring Integrated Backups" on page 154
- "Converting to Integrated Backups" on page 157
- "Consideration for Integrated Backups: Restored Database May Point to Freed Space" on page 160
- "Monitoring Backup Reports for Integrated Backups" on page 160

Overview of Integrated Backup

The integrated-backup method provides a fast, safe, and efficient backup method for DMF. Dumps are sent to a backup filesystem from which the dump data is then migrated to multiple volumes, providing redundancy.

A policy for the backup filesystem must be configured to ensure the following:

- Temporary files produced by the integrated-backup method (which have a project ID of `4136`) are excluded from migration. This is required under the following conditions:
  - If `DUMP_WORKSPACE` is not set
  - If `DUMP_WORKSPACE` is explicitly set to a location inside the backup filesystem (the filesystem that has `USE_AS_DUMP_DEST` set to `ON`)
- All appropriate files are migrated to at least two separate locations. The separate locations can be defined using any combination of volume groups (VGs), media-specific processes (MSPs), or migrate groups (MGs).

The integrated-backup method is the preferred backup method. It applies to basic DMF, DMF in an HA environment, and DMF using the Parallel Data-Mover Option (Parallel DMF).

Contents of the Integrated Backup

An integrated backup consists of the following:

- The `dumps.db` database stored in *HOME_DIR*`/backups` (such as `/home/dmf/backups`) that contains status information about dumps and information about the backup filesystem

- A backup filesystem, such as `/dmf_backups` (defined by the `filesystem` object in the DMF configuration file and labeled with the `DUMP_DESTINATION` parameter), that contains the following:
  - Dumps of the following:
    - Each of the filesystems configured for backup
    - XFS inventory
    - DMF databases
  - Configuration bundle containing the following files:
    - DMF configuration file
    - Tape-driver configuration file
    - `/etc/fstab` file
    - License file
    - Mount information
    - Usage details and UUID of filesystems
    - OpenVault configuration

      Note: OpenVault maintains a history of its configurations. Because backing up the current configuration could create an inconsistent copy, the backup process saves the last snapshot of the OpenVault configuration.

      However, restoring the OpenVault configuration might not always produce the desired result. If the device paths change between the time the configuration was created and the time when the configuration is restored, OpenVault configuration would be invalid. In such cases, manual intervention is required to create a similar configuration from scratch.
  - Disaster-recovery bundle (referred to as the *DR bundle*) containing the following:
    - A copy of the above configuration bundle
    - XFS inventory

- A dump of the backup filesystem
- Disaster-recovery database (containing the DMF database entries that correspond to the backup filesystem)

Note: The files specified by the `FTP_PASSWORD` and `CLOUD_PASSWORD` configuration parameters are not included in the backup. You should keep the contents of those files in a safe location. You can also use the `DUMP_MISC_LOCATIONS` configuration parameter to include them in the dump. See "`taskgroup` Object Parameters" on page 312.

Figure 4-1 describes the backup process.

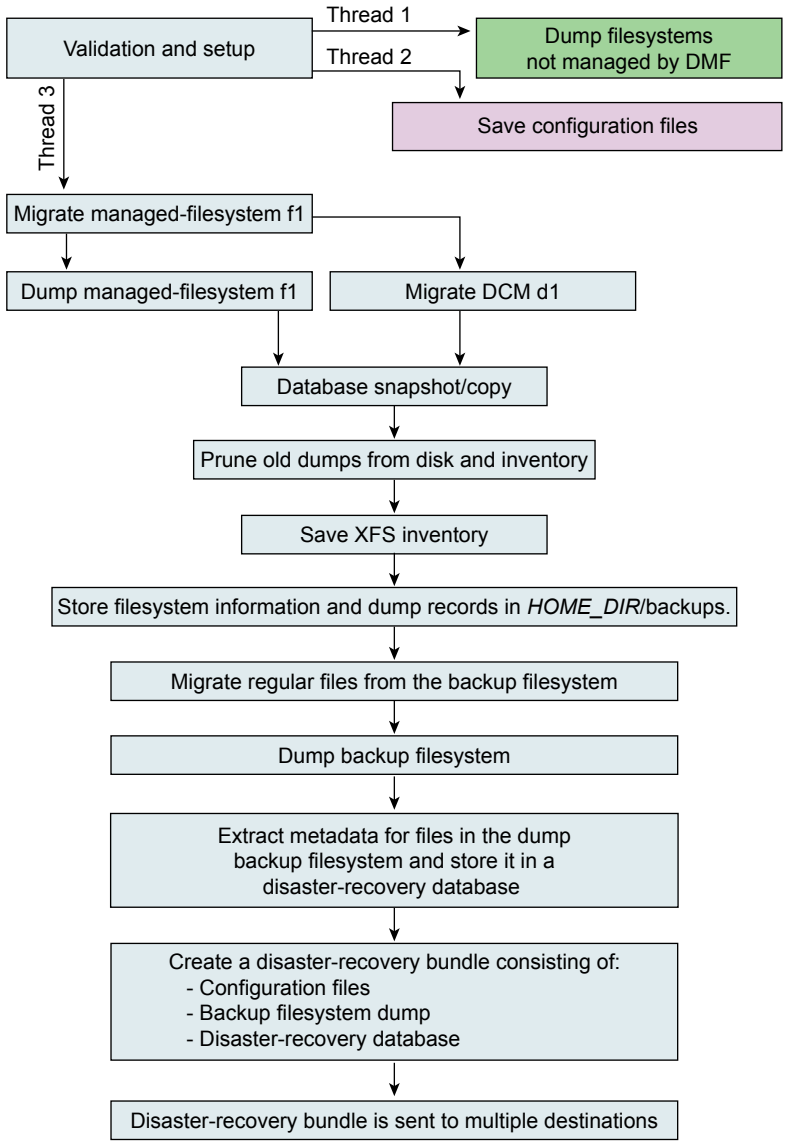

Figure 4-1 DMF Integrated Backups

Figure 4-2 represents the directory structure of the backup filesystem and the location used to store the backup database.

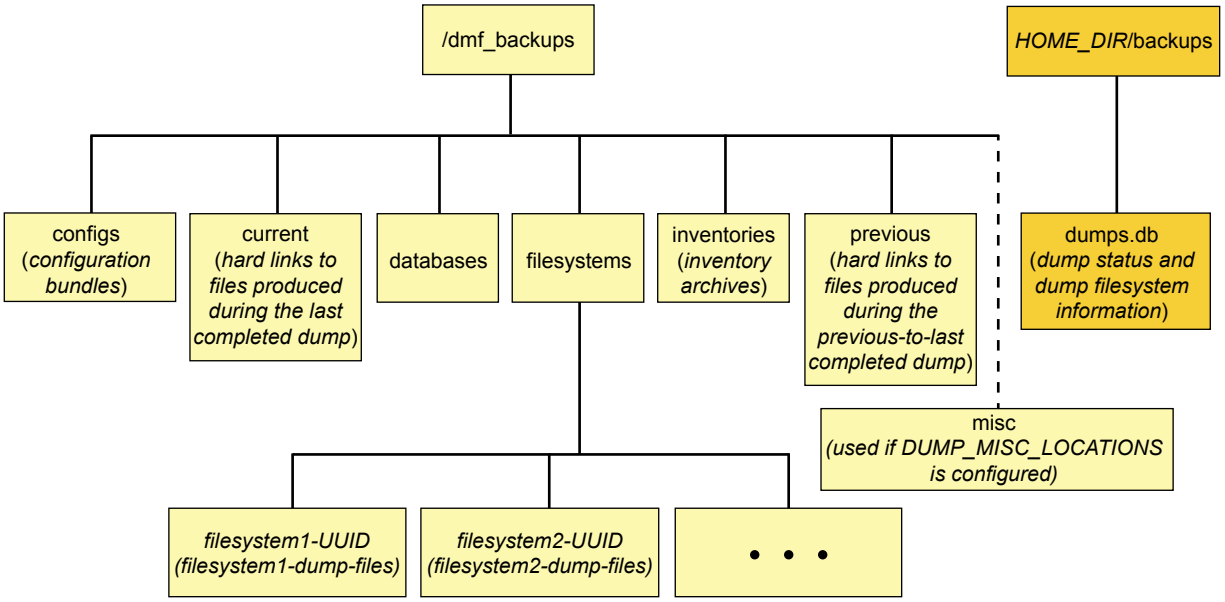

Figure 4-2 Integrated Backup Directory Structures

Size of the Backup Filesystems

Eventually, all of the files in the filesystem that is defined to hold DMF backups (such as `/dmf_backups`) are migrated. The size of the backup filesystem should be such that it is able to hold the files that are created before migration and it should also be able to hold the recalled files during a filesystem restore or disaster-recovery process. As a guideline, the backup filesystem should be large enough to hold the following:

- The DMF databases
- The dumps of filesystems that are configured for backup
- An additional 50-100 GB for other backup components

DR Bundle Locations

The DR bundle must be sent to a safe remote location, at least one of the following:

- A remote mirror specified by the DUMP_DR_MIRRORS configuration parameter. The mirror location requires the following:
  - If using scp(1), the node must permit passwordless login.
  - If using NFS, the mirrored filesystem must be mounted so that the root user is able to access the filesystem with root privileges (that is, with root *squashing* disabled).

Caution: A local mirror is permitted but not recommended because it could be at risk in a disaster, such as a fire. Without a working mirror location, you are in danger of losing the ability to restore filesystems because the /dmf_backups filesystem and its metadata are part of the DR bundle.

- Cloud storage, using a preconfigured site-specific script

For additional safety or testing purposes, the DR bundle may also be sent in an email to the address specified by the DUMP_DR_EMAIL configuration parameter. (The email contains the DR bundle plus information on how to restore a lost bundle from copies on the volume.) If used, you must ensure that you allows a sufficiently large email size limit in the sendmail configuration.

Configuring Integrated Backups

This section discusses the following:

- "Precedence of Objects for Integrated Backups" on page 154
- "Procedure to Configure Objects for Integrated Backups" on page 156

Precedence of Objects for Integrated Backups

Some parameters can be defined in multiple objects in the DMF configuration file. For each filesystem using integrated backups, you can define the following parameters in the dumpgroup object, filesystem object, and taskgroup object:

```
DUMP_COMPRESS
DUMP_MAX_FILESPACE
```

DUMP_MIGRATE_FIRST
DUMP_STREAMS
DUMP_XFSDUMP_PARAMS

In addition, you can define DUMP_FLUSH_DCM_FIRST in the msp object for a DCM MSP and the taskgroup object.

If you define a given parameter in more than one object, the object order of precedence is as follows:

- **Top precedence:**
  - A dumpgroup object that is included in a DUMP_GROUPS parameter of a taskgroup object that applies to integrated backups.
  - An msp object for a DCM MSP that applies to integrated backups
- **Medium precedence:** A filesystem object for a managed filesystem that is configured to be backed up as part of a taskgroup object that applies to integrated backups.
- **Least precedence:** A taskgroup object that applies to integrated backups.

For example, if DUMP_COMPRESS is defined in multiple objects, the value in the dumpgroup object overrides the value in the filesystem object, which overrides the value in taskgroup object.

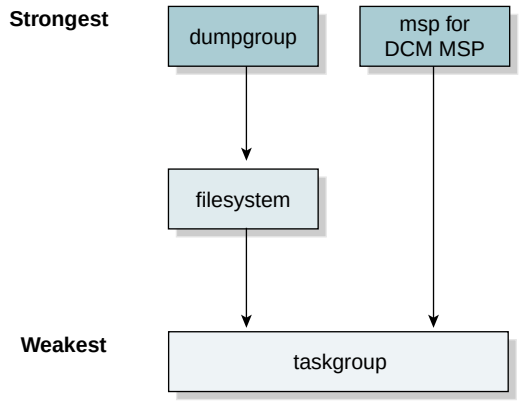

Figure 4-3 Order of Precedence for Dump Parameters

Procedure to Configure Objects for Integrated Backups

To enable integrated backups, do the following:

1. Configure a `filesystem` object for the dump destination.
2. Configure two `policy` objects for the backup filesystem:
   - A `policy` object that defines free-space and migration thresholds
   - A `policy` object that contains the following one or more `SELECT_VG`/`SELECT_MSP` parameters:
     – If `DUMP_WORKSPACE` is not set or if it is explicitly set to a location inside the backup filesystem (the filesystem that has `USE_AS_DUMP_DEST` set to `ON`), then you must prevent the migration of temporary files produced by the integrated-backup method. These temporary files have a project ID of `4136`. To accomplish this, set the **first** `SELECT_VG`/`SELECT_MSP` parameter to the following value:

       ```
       none when projid = 4136
       ```

       Note: The order of `SELECT_VG`/`SELECT_MSP` parameters is important. If this function is required, it must be the first `SELECT_VG`/`SELECT_MSP` parameter in the stanza.

     – At least one `SELECT_VG`/`SELECT_MSP` parameter that defines at least two locations for the migrated data.

   For example:

   ```
   define  backup_vg_policy
           TYPE                    policy
           SELECT_VG               none when projid = 4136
           SELECT_VG               bvg1a bvg1b
   enddef
   ```

3. Configure a single backup `taskgroup` (such as `dump_tasks`) that runs a set of tasks (`run_full_dump.sh`, `run_partial_dump.sh`, and `run_hard_deletes.sh`) on the managed filesystems.
4. If you want to apply different characteristics (such as the number of `xfsdump` streams or the compression type) to specific managed filesystems, you can do so by setting various `DUMP_*` parameters in the `filesystem` or `dumpgroup` objects

for those specific filesystems; see "Precedence of Objects for Integrated Backups" on page 154.

5. *(HA or Parallel DMF only)* Copy the certificate file to all DMF nodes.

   In an environment using HA or Parallel DMF, the backup process will collect information from all potential DMF servers and all parallel data-mover nodes using a `xinetd`-based communication mechanism that uses SSL certificates for authentication. To achieve this, you must run the `ov_makecert`(8) command on the active DMF server and manually copy the resulting certificate file to the passive DMF server (in an HA configuration) and any parallel data-mover nodes. For more information, see the `ov_makecert`(8) command.

For details, see:

- "Precedence of Objects for Integrated Backups" on page 154
- "`taskgroup` Object" on page 304
- "`dumpgroup` Object" on page 342
- "`filesystem` Object" on page 350
- "`policy` Object" on page 364

Converting to Integrated Backups

If you want to convert an existing system to integrated backups, do the following:

1. Prepare for the conversion by executing the following command, depending upon your current nonintegrated backup method:

   - Disk backups:

     # **`/usr/sbin/dmconvertbackup`**

   - Tape backups (or a mix of disk and tape):

     # **`/usr/sbin/dmconvertbackup`** [**`-V`** ***`VG_or_AG`*** [**`-l`** ***`LS`***]]

     where:

     `-l` *LS* — Specifies the library server to which freed-up tapes will be added for the specified VG or AG

`-V` *VG_or_AG*	Specifies the VG or AG to which freed-up tapes will be added.

The first time that you execute `dmconvertbackup`, it does the following:

a. Adds a daily `cron`(8) job that performs prune-only runs of existing nonintegrated disk and tape backup tasks.

b. Saves a copy of the DMF configuration file in the *HOME_DIR*`/convertbackup` directory. Arguments passed to dmconvertbackup are also stored in this copy of the DMF configuration.

c. Saves a copy of the files specified by the `DUMP_TAPES` configuration parameter (for nonintegrated tape-based backups only) in the *HOME_DIR*`/convertbackup` directory.

d. Prunes dumps from nonintegrated backups according to their respective `DUMP_RETENTION` values.

e. Optionally returns any freed-up tapes to the specified volume group (VG) or allocation group (AG) in the specified library server (LS).

Note: Subsequent executions of this command perform step 1d through step 1e until there are no dumps from nonintegrated backups in the XFS dump inventory. At that point, the command removes the daily `cron` job that calls this command and removes the *HOME_DIR*`/convertbackup` directory.

For example:

- To install a `cron` job that prunes nonintegrated disk and tape dumps based on the `DUMP_RETENTION` values configured in their respective task groups:

  # **`dmconvertbackup`**

- To install a cron job as described above and also return tapes thus freed up to AG `myag` in LS `myls`:

  # **`dmconvertbackup -V myag -l myls`**

2. Make changes to the DMF configuration file. For example:

   a. Delete the nonintegrated `taskgroup` objects that pertain to backups.

b. Create a new `taskgroup` object for the integrated backups. This new object must have a unique name that differs from the nonintegrated `taskgroup` names. For example, you could use `integrated_dumptasks`.

Note: You must use a new, unique name for the integrated backup `taskgroup` because the command uses the names of the nonintegrated backup `taskgroup` objects to determine if there is a nonintegrated backup still to be pruned from the `xfsdump` inventory entries.

If you want to use dump parameter values that differ between filesystems, you can include those parameters in the `filesystem` or `dumpgroup` objects. See "Precedence of Objects for Integrated Backups" on page 154.

c. Configure two VGs sharing one AG (specified above)

d. Define `backup_vg_policy` to exclude temporary files produced by the integrated-backup method (which have project ID `4136`) and to use two VG locations for the migrated files. For example:

```
define backup_vg_policy
    TYPE                policy
    SELECT_VG           none when projid = 4136
    SELECT_VG           bg_vg1 bg_vg2
```

e. Use a managed filesystem as the backup filesystem. This filesystem is used as the `DUMP_DESTINATION` value for the integrated-backup `taskgroup` object. This filesystem object must set `USE_AS_DUMP_DEST` to `ON` and use the `backup_vg_policy` defined in the above step. The following is an example excerpt:

```
define  /dmf_backups
    TYPE                filesystem
    MIGRATION_LEVEL     auto
    POLICIES            backup_space_policy backup_vg_policy
    USE_AS_DUMP_DEST    on
enddef

define integrated_dumptasks
    TYPE                taskgroup
    DUMP_DESTINATION    /dmf_backups
    DUMP_DR_MIRRORS     root@mymachine:/tmp/mymirrors
```

```
    DUMP_DR_EMAIL       admin@company.com
enddef
```

See also:

- "`taskgroup` Object Example for Integrated Backups" on page 332
- "`dumpgroup` Object Example for Integrated Backups" on page 345
- "`filesystem` Object Example for Integrated Backups" on page 362

Consideration for Integrated Backups: Restored Database May Point to Freed Space

If DMF database is corrupt or lost and must be restored from a period older than *HFREE_TIME*, it is possible that the entries in the older database copy point to the space that has already been freed. This could lead to an incomplete DMF recovery and partial data loss. Although you would normally immediately notice a problem requiring disaster recovery actions, it is possible for a problem to go undetected for some time. Therefore, you should set `HFREE_TIME` to a value that is large enough to accommodate the use of an older DR bundle. Assuming that full backups are done weekly, SGI recommends setting `HFREE_TIME` to `1209600` (2 weeks).

Note: The value of the `MERGE_THRESHOLD` parameter for the `volumegroup` object and the `THRESHOLD` parameter for the `taskgroup` object determine when sparsely occupied tapes are merged. Any successful tape merge effectively invalidates the previous DMF database backups and DR bundles; therefore, setting an appropriate `HFREE_TIME` helps keep enough DMF database backups and DR bundles valid for disaster recovery.

Monitoring Backup Reports for Integrated Backups

You should monitor the backup report daily to verify that there is sufficient storage space for future backups. Integrated backups regularly prune older backups based on `DUMP_RETENTION` time, but to free up space after pruning, it is important to run hard deletes followed by periodic volume merges.

Other Backup Methods

This section discusses nonintegrated disk backups and nonintegrated tape backups:

- "Report Monitoring for Nonintegrated Tape Backups" on page 161
- "Storage Used by an FTP MSP or a Standard Disk MSP for Nonintegrated Backups" on page 161
- "Filesystems Used by a DCM MSP for Nonintegrated Backups" on page 161
- "DMF's Private Filesystems and Nonintegrated Backups" on page 163

Report Monitoring for Nonintegrated Tape Backups

You should monitor the tape backup report daily to verify that there are sufficient tapes available for future backups. If it turns out that a given day requires multiple backup tapes for the set of backups for that day's backup, you must empty previously used backup tapes or add more backup tapes.

Storage Used by an FTP MSP or a Standard Disk MSP for Nonintegrated Backups

If you are depending on an FTP MSP or a standard disk MSP to provide copies of your offline files in order to safeguard your data, then they should also be backed up.

If you use them just to hold extra copies for convenience or to speed data access, they need not be backed up. But you should consider how you would handle their loss. You would probably need to remove references to lost copies from the DMF daemon database, using `dmdadm`, which can only be done when the daemon is not running.

Filesystems Used by a DCM MSP for Nonintegrated Backups

A DCM MSP differs from a disk MSP in that it uses DMAPI to manage the files. It will not operate properly if the files are reloaded by a package that cannot also restore the DMAPI information associated with each file.

Note: For simplicity, this discussion assumes that the site wishes to keep two copies of migrated files at all times to guard against media problems. (Keeping only one copy is considered risky, and keeping more than two copies is frequently impractical.)

The DCM MSP can have one of the following configurations:

- A DCM MSP may be holding an extra copy of files in addition to the normal number of tape-based or MAID-based copies. That is, after the initial migration has completed, there will be two lower-tier copies and a third in the cache. The DCM MSP may easily remove this third copy from the cache after some period of time, just leaving two lower-tier copies. With this configuration, there is normally no need to back up the cache filesystem.
- The initial migration could result in one cache copy and one on lower tier. Later on, when the cache has to be flushed, a second lower-tier copy is written by the DCM MSP before the cache-resident one is deleted. If the file is hard-deleted before the cache flushes, the second lower-tier copy will never be made, thereby saving time and lower-tier space. The tradeoff is that cache-flushing is slower and the cache filesystem should be backed up; otherwise a media problem in conjunction with a disk failure would result in data loss. With this configuration, the cache filesystem should be backed up. Otherwise, the loss of the cache disk could leave you with just one copy of data on a lower tier. This is considered to be risky.

For both configurations, any backups require the use of a backup package that is aware of DMF (as listed in Appendix D, "Third-Party Backup Package Configuration" on page 767) to back up the cache.

To use `run_full_dump.sh` or `run_partial_dump.sh` to back up any of these filesystems, include the pathname of its mountpoint in the `DUMP_FILE_SYSTEMS` parameter.

DMF's Private Filesystems and Nonintegrated Backups

The following DMF private filesystems do not require a backup package that is aware of DMF when using nonintegrated backups:

HOME_DIR
JOURNAL_DIR
SPOOL_DIR
TMP_DIR
CACHE_DIR
MOVE_FS

Take care when backing up the databases in *HOME_DIR* if there is any DMF activity going on while the backup is underway, due to the risk of making the copy of the database while it is being updated. A safe technique is to take a snapshot of the databases with `dmsnap` and back up the snapshot. The `run_full_dump.sh` or `run_partial_dump.sh` script does this automatically.

The journal files in *JOURNAL_DIR* should also be backed up if you keep older snapshots of the databases that may have to be reloaded and brought up-to-date with `dmdbrecover`. Preferably, journals should be backed up when DMF activity (apart from recalls) is minimal. The `run_full_dump.sh` and `run_partial_dump.sh` scripts and parameters `DUMP_MIGRATE_FIRST` and `DUMP_FLUSH_DCM_FIRST` help achieve this by processing any queued up migration requests immediately before starting the backup.

SPOOL_DIR contains log files that may be of use for problem diagnosis, as well as history files controlling things like media error recovery and reporting scripts. The loss of these files will not endanger user data, although DMF may act a little differently for a while until it reestablishes them. Back up *SPOOL_DIR* if you can.

The *TMP_DIR*, *CACHE_DIR*, and *MOVE_FS* filesystems do not require backup.

To use `run_full_dump.sh` or `run_partial_dump.sh` to back up any of these filesystems, simply include the pathnames of their mountpoints in the `DUMP_FILE_SYSTEMS` parameter.

Installing and Configuring the DMF Environment

This chapter discusses the following:

- "Overview of the Installation and Configuration Steps" on page 165
- "Installation and Configuration Considerations" on page 169
- "Starting and Stopping the DMF Environment" on page 181
- "Using Out-of-Library Tapes" on page 184
- "Customizing DMF" on page 186
- "Importing Data From Other HSMs " on page 189

Overview of the Installation and Configuration Steps

To install and configure the DMF environment, perform the following steps:

Note: Also see:

- *COPAN MAID for DMF Quick Start Guide*
- *SGI 400 VTL for DMF Quick Start Guide*

Procedure 5-1 Configuring the DMF Environment

1. Read "Installation and Configuration Considerations" on page 169.
2. Ensure that the DMF server is running the `xinetd`(8) daemon. If it has been disabled, you must reenable it at boot time via the following command:

   ```
   dmfserver# chkconfig xinetd on
   ```

   If `xinetd` is not running, you can start it immediately via the following command:

   ```
   dmfserver# service xinetd start
   ```

   See also "Set the `xinetd` `tcpmux` `instances` Parameter Appropriately" on page 108.

3. Install the DMF server software (which includes the software for TMF and OpenVault) according to the instructions in the *SGI InfiniteStorage Software Platform* release note and any late-breaking caveats posted to the online download page for the release:

   https://support.sgi.com

   See "ISSP DMF Software" on page 169.

4. Determine the DMF drive groups that you want to use.

5. Start the `ident` service on the DMF server:

   - RHEL:

     a. Configure the `auth` service to start when the `xinetd` service is started:

        ```
        dmfserver-rhel# chkconfig auth on
        ```

     b. Modify the `/etc/xinetd.d/pidentd` file and add the `only_from` option to ensure that only the PostgreSQL database server will connect and use `pidentd`. The PostgreSQL server is the `hostname` value for the DMF server (or its IP address); in an HA environment, use the virtual hostname.

        For example, in a non-HA environment if the DMF server `hostname` is `dmfserver`:

        ```
        only_from = dmfserver
        ```

        For example, in an HA environment if the virtual hostname used for the active DMF server is `virtual-server`:

        ```
        only_from = virtual-server
        ```

     c. Verify that the `server_args` parameter in the `/etc/xinetd.d/auth` file **does not** contain `-E` in its argument list.

        If `-E` is present, edit the file to remove it and then restart the `xinetd` service:

        ```
        dmfserver-rhel# service xinetd restart
        ```

   - SLES:

     a. Configure the `pidentd` service to start when the `xinetd` service is started:

        ```
        dmfserver-sles# chkconfig pidentd on
        ```

b. Modify the `/etc/xinetd.d/auth` file and add the `only_from` option to ensure that only the PostgreSQL database server will connect and use `auth`. The PostgreSQL server is the `hostname` value for the DMF server (or its IP address); in an HA environment, use the virtual hostname.

   For example, in a non-HA environment if the DMF server `hostname` is `dmfserver`:

   ```
   only_from = dmfserver
   ```

   For example, in an HA environment if the virtual hostname for the DMF server is `virtual-server`:

   ```
   only_from = virtual-server
   ```

c. Verify that the `server_args` parameter in the file `/etc/xinetd.d/pidentd` **does not** contain `-E` in its argument list.

   If `-E` is present, edit the file to remove it and then restart the `xinetd` service:

   ```
   dmfserver-sles# service xinetd restart
   ```

6. Configure the TMF or OpenVault mounting service (if used) according to the following documentation:

   - *TMF 6 Administrator Guide for SGI InfiniteStorage*
   - *OpenVault Administrator Guide for SGI InfiniteStorage*

7. Determine how you want to complete periodic maintenance tasks. See "Automated Maintenance Tasks" on page 175.

8. Make and mount the filesystems required for the DMF administrative directories. See "Configure Filesystems and Directories Appropriately for DMF" on page 94.

9. Install the DMF license (and optional DMF Parallel Data-Mover Option license) on the primary DMF server and the passive DMF server (if applicable). See Chapter 2, "DMF Licensing" on page 67 and "Managing Licenses and Data Capacity with DMF Manager" on page 203.

   Note: Nodes running DMF client software do not require a DMF license.

10. Create or modify your configuration file and define objects for the following:

- Pathname and file size parameters necessary for DMF operation (the `base` object)
- DMF daemon
- Daemon maintenance tasks
- Filesystems
- Automated space management
- Media-specific process (MSP) or library server (LS)
- MSP/LS maintenance tasks

See "Configuring DMF with DMF Manager" on page 210.

Also see " Object Definitions" on page 272.

11. Verify the configuration by selecting the following in DMF Manager, select the following:

    Overview
    > Configuration ...
    > Validate Current Configuration

    If there are errors, fix them and repeat the validation until there are no errors.

12. If you are using the DMF Parallel Data-Mover Option, see "Parallel DMF Configuration Procedure" on page 485 and the *SGI InfiniteStorage Software Platform* release note.

13. Start the DMF environment. See "Starting and Stopping the DMF Environment" on page 181.

14. If you want to install the DMF client packages on other systems, see the *SGI InfiniteStorage Software Platform* release note and the client installation instructions.

To administer and monitor DMF, see Chapter 6, "DMF Manager" on page 191.

Installation and Configuration Considerations

This section discusses the configuration considerations that will affect your system:

- "ISSP DMF Software" on page 169
- "Mounting Service Considerations" on page 170
- "Inode Size Configuration" on page 171
- "Daemon Database Record Length" on page 173
- "Interprocess Communication Parameters" on page 175
- "Automated Maintenance Tasks" on page 175
- "Networking Considerations for Parallel Data-Mover Option" on page 176
- "Passwordless SSH Configuration for DMF" on page 176
- "Suppressing RSCN on Host Port HBAs" on page 177
- "QLogic® Fibre Channel Switch" on page 178
- "IPv6" on page 181

ISSP DMF Software

The ISSP release includes the following DMF software:

- **DMF Server**, which provides:
  - The full set of DMF server functionality, including the DMF daemon, infrastructure, user and administrator commands, and all man pages. This applies to SGI x86_64 servers running the operating system as specified in the ISSP and DMF release notes. You should install this software only on those machines that can be the DMF server.
  - Client installers, which download the client software onto the DMF server so that you can later transfer the DMF client software to the DMF client nodes. The client packages are installed along with their installation instructions on the DMF server in the following directory:

    `/opt/dmf/client-dist/`*`DMFVERSION`*`/`*`clientOS`*`/`*`clientarchitecture`*

The client software contains a limited set of DMF functionality. This applies to all supported operating systems. You should install this software on machines from which you want to give users access to DMF operations, such as migrating and recalling files..

- **DMF Parallel Data-Mover**, which provides the infrastructure for parallel data-mover nodes to move data offline and retrieve it, plus the required underlying CXFS client-only software.

Only one group of software can be installed on a given machine.

Mounting Service Considerations

Mounting services are available through OpenVault or the Tape Management Facility (TMF)

The LS checks the availability of the mounting service when it is started and after each occurrence in which an LS data-mover process was unable to reserve its drive. The data-mover process may be either:

- A *write child* that migrates data to secondary storage
- A *read child* that recalls data from secondary storage

If the mounting service is unavailable, the LS does not start any new child processes:

- For OpenVault, the LS sends an e-mail message to the administrator, asking that OpenVault be started. It then periodically polls OpenVault until it becomes available, at which time child processes are again allowed to run.
- For TMF, the LS attempts to initiate `tmdaemon` if it is not up (based on the exit status of `tmstat`) and waits until a TMF device in the `configuration pending` state is configured up before it resumes processing. If TMF cannot be started or if no devices are configured up, the LS sends e-mail to the administrator and polls TMF until a drive becomes available.

You can use `MAX_MS_RESTARTS` to configure the number of automatic restarts.

See also Chapter 9, "Mounting Service Configuration Tasks" on page 491

Inode Size Configuration

In managed filesystems and disk cache manager (DCM) MSP *STORE_DIRECTORY* filesystems, DMF state information is kept within a filesystem structure called an *extended attribute*.

Extended attributes can be either inside the inode or in attribute blocks associated with the inode. DMF runs much faster when the extended attribute is inside the inode, because this minimizes the number of disk references that are required to determine DMF information. In certain circumstances, there can be a large performance difference between an inode-resident extended attribute and a non-resident extended attribute.

The size of inodes within a filesystem impacts how much room is available inside the inode for storing extended attributes. Smaller inode sizes have much less room available for attributes. Likewise, the legacy inode attribute format (`-i attr=1` option to `mkfs.xfs`) results in less available extended attribute space than does the current default format (`-i attr=2` option).

SGI recommends that you configure your filesystems so that the extended attribute is always inode-resident. Whenever both 256-byte and 512-byte inode sizes will work, you should use the 256-byte inode size (`-i size=256` option to `mkfs.xfs`) because the inode scans will be up to twice as fast. SGI also highly recommends that you use `attr2` (`-i attr=2` option) when possible if it allows 256-byte inode sizes to be used.

For optimal performance, you should create any DCM MSP filesystems with 256-byte inode sizes and `attr2` attribute format. (For other filesystems for DMF administrative directories, the inode size does not matter.) For best performance, you should use 512-byte inode sizes for managed filesystems only under the following circumstances:

- If users require other XFS attributes such as ACLs or other user-specified attributes
- If the filesystem will have large numbers of partial-state files with more partial-state regions than will fit in a 256-byte inode

If you must have a 512–byte inode size for a managed filesystem, you can do so by using the Linux `mkfs.xfs` command with the `-i size=512` option. (Filesystems that already exist must be backed up, recreated, and restored.)

Table 5-1 summarizes the relationship among the inode size, `attr` type, and file regions.

Table 5-1 Default Maximum File Regions for XFS and CXFS Filesystems

Size of inode	`attr` Type	Default Maximum Number of File Regions
256	`1`	(Not recommended)
256	`2`	2
512 or greater	`1`	8
512 or greater	`2`	11

For more information about setting the inode size and the `attr` type, see the `mkfs.xfs`(8) and `mount`(8) man pages.

Daemon Database Record Length

A daemon database entry contains one or more fixed-length records:

- The base record (`dbrec`), which consists of several fields, including the `path` field
- Zero or more path segment extension (`pathseg`) records

If the value that is returned to the daemon by the MSP/LS (such as the pathname resulting from the `NAME_FORMAT` value template in a cloud, disk, or FTP `msp` object) can fit into the `path` field of the daemon's `dbrec` record, DMF does not require `pathseg` records. If the MSP/LS supplies a path value that is longer than the `path` field, DMF creates one or more `pathseg` records to accommodate the extra space.

The default size of the `path` field of the `dbrec` is 34 characters. This size allows the default paths returned by `dmatls`, `dmcloudmsp`, `dmdskmsp`, and `dmftpmsp` to fit in the `path` field of `dbrec` as long as the user name portion of the `dmcloudmsp`, `dmdskmsp`, or `dmftpmsp` or default path (*username/bit_file_identifier*) is 8 characters or fewer. If you choose to use a value for `NAME_FORMAT` that results in longer pathnames, you may want to resize the `path` field in `dbrec` in order to increase performance.

The default size of the path field in the `pathseg` record is 64. For MSP path values that are just slightly over the size of the `dbrec` path field, this will result in a large amount of wasted space for each record that overflows into the `pathseg` record. The ideal situation would be to have as few `pathseg` records as possible, because retrieving `pathseg` records slows down the retrieval of daemon database records.

The size of the path field in the daemon `dbrec` record can be configured at any time before or after installation. (The same holds true for any installation that might be using `dmcloudmsp`, `dmdskmsp`, or `dmftpmsp` with a different path-generating algorithm or any other MSP that supplies a path longer than 34 characters to the daemon.)

Procedure 5-2 Configuring the Daemon Database Record Length

The steps to configure the daemon database entry length are as follows:

1. If `dmfdaemon` is running, ensure that DMF is stopped. In an HA environment, see *High Availability Guide for SGI InfiniteStorage*. In a non-HA environment, execute the following:

   ```
   dmfserver# service dmf stop
   ```

Caution: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage*.

2. If a daemon database already exists, perform the following commands:

```
dmfserver# cd HOME_DIR/daemon
dmfserver# dmdump -c . > textfile
dmfserver# cp dbrec* pathseg* dmd_db.dbd backup_dir
dmfserver# rm dbrec* pathseg* dmd_db.dbd
```

   Where:

   - *HOME_DIR* is the value of `HOME_DIR` returned by the `dmconfig base` command
   - *textfile* is the name of a file that will contain the text representation of the current daemon database
   - *backup_dir* is the name of the directory that will hold the old version of the daemon database

3. Change to the `rdm` directory:

```
dmfserver# cd /usr/lib/dmf/rdm
```

4. Back up the `dmd_db.dbd` and `dmd_db.ddl` files that reside in `/usr/lib/dmf/rdm`. This will aid in disaster recovery if something goes wrong.

5. Edit `dmd_db.ddl` to set the new `path` field lengths for the `dbrec` and/or `pathseg` records.

6. Regenerate the new daemon database definition, as follows:

```
dmfserver# /usr/lib/dmf/support/dmddlp -drsx dmd_db.ddl
```

7. Back up the new versions of `dmd_db.dbd` and `dmd_db.ddl` for future reference or disaster recovery.

8. If the daemon database was backed up to text (to *textfile* in step 2), enter the following commands:

   ```
   dmfserver# cd HOME_DIR/daemon
   dmfserver# dmdadm -u -c "load textfile"
   ```

9. If the daemon was running in step 1, ensure that the `dmf` service is started. In a non-HA environment, execute the following:

   ```
   dmfserver# service dmf start
   ```

Interprocess Communication Parameters

Ensure that the following interprocess communication kernel configuration parameters are set equal to or greater than the default before running DMF:

```
MSGMAX
MSGMNI
```

For more information, execute `info ipc` and see the `sysctl`(8) and `msgop`(2) man pages.

Automated Maintenance Tasks

DMF lets you configure parameters for completing periodic maintenance tasks such as the following:

- Making backups (full or partial) of managed filesystems to tape or disk
- Making backups of DMF databases to disk
- Removing old log files and old journal files
- Monitoring DMF logs for errors
- Monitoring the status of volumes in LSs
- Running hard deletes
- Running `dmaudit`(8)
- Merging volumes that have become sparse (and stopping this process at a specified time)

Each of these tasks can be configured in the DMF configuration file (`/etc/dmf/dmf.conf`) through the use of `TASK_GROUPS` parameters for the DMF daemon and the LS. The tasks are then defined as objects.

For each task you configure, a time expression defines when the task should be done and a script file is executed at that time. The tasks are provided in the `/usr/lib/dmf` directory.

The automated tasks are described in "`taskgroup` Object" on page 304.

Networking Considerations for Parallel Data-Mover Option

The parallel data-mover nodes communicate with the DMF and OpenVault servers over the network. By default, they use the IP addresses that are associated with the system hostnames. Additionally, depending on your configuration, it is possible that socket merging can occur between hosts. By default, this feature uses the same network as other DMF communication traffic.

It is possible to configure DMF to use an alternative network for general communication between DMF nodes as well as an alternative network for socket merges. See "`node` Object" on page 294.

If you use an alternative network for DMF communication, the OpenVault server must listen on the same network; in this case, the name you specify for the initial OpenVault prompt (that asks you to supply the name where OpenVault will be listening) will be different from the hostname of the DMF server. See comment 2 in "Configure the OpenVault Server" on page 492.

Passwordless SSH Configuration for DMF

If configured, DMF will use passwordless secure shell (SSH) to do the following.

- Transfer a copy of nonintegrated disk backups to one or more remote directories (using the optional `DUMP_MIRRORS` DMF configuration file)
- Simplify the use of the `dmatsnf`(8) and `dmatread` (8) commands to verify the integrity of the library server (LS) volumes and recover data from them for a configuration where not all volumes are mountable on the DMF server
- When in an active-active HA configuration with parallel data-mover nodes, DMF Manager can represent the status of both mover nodes

- Gather troubleshooting data via the dmcollect(8) command from all potential DMF servers and parallel data-mover nodes.

You must set up SSH keys so that the local root user can log in to the remote host as a remote user without a password. Do the following:

1. Generate RSA authentication keys for the root user on the DMF server, if the keys do not already exist. Be sure that you **do not** enter a passphrase when prompted (just press Enter).

   ```
   dmfserver# ssh-keygen -t rsa
   Generating public/private rsa key pair.
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in /root/.ssh/id_rsa.
   Your public key has been saved in /root/.ssh/id_rsa.pub.
   ...
   ```

2. Install the identity information on all nodes on which passwordless SSH access is required. For example:

   - To provide access for the DMF server on two parallel data-mover nodes pdm1 and pdm2:

     ```
     dmfserver# ssh-copy-id -i ~/.ssh/id_rsa.pub root@pdm1
     dmfserver# ssh-copy-id -i ~/.ssh/id_rsa.pub root@pdm2
     ```

   - To provide access when using a remote host that has directories in which DMF will place a copy of nonintegrated disk backups (DUMP_MIRRORS):

     ```
     dmfserver# ssh-copy-id -i ~/.ssh/id_rsa.pub user@remotehost
     ```

See the ssh-keygen(1) and ssh-copy-id(1) man pages for details.

Suppressing RSCN on Host Port HBAs

To avoid cluster problems when one node in the cluster reboots, enable (turn ON) registered state change notification (RSCN) suppression on host port HBAs. (HBAs do not communicate with HBAs, therefore a port registration request does not need to occur between HBAs.) Use the portcfgshow command to display the current settings.

Note: Proper zoning will ensure that multiple host ports are not contained within the same zone.

Use the following command on the switch:

```
switch> portcfg rscnsupr [Slot/]Port[-Range] --enable|--disable
```

For example, suppose that ports 4 through 7 go from the switch to nodes in the cluster. You would enter the following to enable RSCN suppression for ports 4 through 7:

```
switch> portcfg rscnsupr 4-7 --enable

switch> portcfgshow

Locked L_Port     .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
Locked G_Port     .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
Disabled E_Port   .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
ISL R_RDY Mode    .. .. .. ..   .. .. .. ..   .. .. .. ..   .. .. .. ..
RSCN Suppressed   .. .. .. ..   ON ON ON ON   .. .. .. ..   .. .. .. ..
Persistent Disable.. ON ON ON   .. ON ON ON   ON ON ON ON   ON .. .. ..
NPIV capability   ON ON ON ON   ON ON ON ON   ON ON ON ON   ON ON ON ON
```

QLogic® Fibre Channel Switch

All QLogic Fibre Channel (FC) switches contained within the SAN fabric must have the appropriate QLogic firmware installed.

For more information, see the QLogic *SANbox2-64 Switch Management User's Guide.*

Caution: The `admin` state is required for I/O fencing. To avoid interference with fencing, release `admin` mode as soon as possible. Do not leave `admin` mode sessions open.

The default port configuration on a QLogic 9200 FC switch is not compatible with the DMF environment. To use the appropriate port configuration, change the following parameters:

`LinkSpeed`	Set to the appropriate value, such as `2` for 2 GB/s. (In some cases, `Auto` does not function properly.)

`PortType`	Enter the appropriate type, usually `F`. (You cannot use the `GL` autonegotiated mode.)
`NoClose`	Set to `True` to prevent the Fibre Channel circuit from shutting down during a host reboot.
`IOStreamGuard`	Set to `Enable` only on ports connected to initiator devices (HBAs). This feature suppresses the generation of RSCN messages on a port basis. (You cannot use `Auto` mode because most HBAs cannot negotiate this.)

To modify these parameters, use the `admin` command. For example, for a port connected to an SGI UV® 100 system:

```
SANbox #> admin start

SANbox (admin) #> config edit
  The config named default is being edited.

SANbox (admin-config) #> set config port 31


  A list of attributes with formatting and current values will follow.
  Enter a new value or simply press the ENTER key to accept the current value.
  If you wish to terminate this process before reaching the end of the list
  press 'q' or 'Q' and the ENTER key to do so.

  Configuring Port Number: 31
  -------------------------

  AdminState       (1=Online, 2=Offline, 3=Diagnostics, 4=Down) [Online]
  LinkSpeed        (1=1Gb/s, 2=2Gb/s, 4=4Gb/s, A=Auto)          [Auto  ]  2
  PortType         (GL / G / F / FL / Donor)                    [GL    ]  F
  SymPortName      (string, max=32 chars)                       [Port31]  UV100
  ALFairness       (True / False)                               [False ]
  DeviceScanEnable (True / False)                               [True  ]
  ForceOfflineRSCN (True / False)                               [False ]
  ARB_FF           (True / False)                               [False ]
  InteropCredit    (decimal value, 0-255)                       [0     ]
  ExtCredit        (dec value, increments of 15, non-loop only) [0     ]
  FANEnable        (True / False)                               [True  ]
  AutoPerfTuning   (True / False)                               [True  ]
  MSEnable         (True / False)                               [True  ]
```

```
  NoClose          (True / False)                          [False ]  True
  IOStreamGuard    (Enable / Disable / Auto)               [Auto  ]  Enable
  PDISCPingEnable  (True / False)                          [True  ]

  Finished configuring attributes.
  This configuration must be saved (see config save command) and
  activated (see config activate command) before it can take effect.
  To discard this configuration use the config cancel command.
  ....

SANbox (admin-config) #> config save
  The config named default has been saved.

SANbox (admin) #> config activate

  The currently active configuration will be activated.
  Please confirm (y/n): [n] y

SANbox (admin) #> admin end

SANbox #> show config port 31

  Configuration Name: default
  -------------------

  Port Number: 31
  ------------
  AdminState         Online
  LinkSpeed          2Gb/s
  PortType           F
  SymbolicName       UV100
  ALFairness         False
  DeviceScanEnabled  True
  ForceOfflineRSCN   False
  ARB_FF             False
  InteropCredit      0
  ExtCredit          0
  FANEnabled         True
  AutoPerfTuning     True
  MSEnabled          True
  NoClose            True
```

```
IOStreamGuard      Enabled
PDISCPingEnabled   True
```

IPv6

To implement DMF with IPv6, do the following:

- Add `IPv6` to the `flags` parameter in the `/etc/xinetd.d/tcpmux` file:

  ```
  flags        = IPv6 IPv4
  ```

- Change to the `/etc/vsftpd.conf` file listen parameter to specify IPv6:

  ```
  #listen=YES
  listen_ipv6=YES
  ```

Note: You should comment-out or remove the `listen` parameter, because it is mutually exclusive with the `listen_ipv6` parameter.

Starting and Stopping the DMF Environment

This section discusses the following:

- "Automatic Start After Reboot" on page 182
- "Preventing Automatic Start After Reboot" on page 182
- "Explicit Start" on page 183
- "Explicit Stop" on page 183

For more information about the mounting services, see:

- *TMF 6 Administrator Guide for SGI InfiniteStorage*
- *OpenVault Administrator Guide for SGI InfiniteStorage*

Caution: In an HA environment, procedures differ. For example, you must first remove HA control of the resource group before stopping DMF and the mounting service. See *High Availability Guide for SGI InfiniteStorage*.

Automatic Start After Reboot

Note: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage*

To enable automatic startup of the DMF environment, execute the following `chkconfig`(8) commands as `root` on indicated nodes in a non-HA environment:

1. DMF server:

```
dmfserver# chkconfig pcp on
dmfserver# chkconfig pcp-storage on (optional)
dmfserver# chkconfig tmf on        (if TMF)
dmfserver# chkconfig openvault on  (if OpenVault)
dmfserver# chkconfig dmf on
dmfserver# chkconfig dmfman on     (optional)
dmfserver# chkconfig dmfsoap on    (optional)
```

2. Parallel data-mover nodes:

```
mover# chkconfig openvault on
mover# chkconfig dmf_mover on
```

Preventing Automatic Start After Reboot

Note: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage.*

To prevent automatic startup of the DMF environment, execute the following `chkconfig`(8) commands as `root` on the indicated nodes in a non-HA environment:

1. DMF server:

```
dmfserver# chkconfig dmfsoap off
dmfserver# chkconfig dmfman off
dmfserver# chkconfig dmf off
dmfserver# chkconfig openvault off    (if OpenVault)
dmfserver# chkconfig tmf off          (if TMF)
```

2. Parallel data-mover nodes:

```
mover# chkconfig dmf_mover off
mover# chkconfig openvault off
```

Explicit Start

Note: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage.*

To start the DMF environment daemons explicitly, execute the following on the indicated nodes in a non-HA environment:

1. DMF server:

```
dmfserver# service pcp start
dmfserver# service pcp-storage start (optional)
dmfserver# service tmf start         (if TMF)
dmfserver# service openvault start   (if OpenVault)
```

2. Parallel data-mover nodes:

```
mover# service openvault start
```

3. DMF server:

```
dmfserver# service dmf start
```

4. Parallel data-mover nodes:

```
mover# service dmf_mover start
```

5. DMF server:

```
dmfserver# service dmfman start      (optional)
dmfserver# service dmfsoap start     (optional)
```

Explicit Stop

Note: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage.*

To stop the DMF environment daemons explicitly, execute the following on the indicated nodes in a non-HA environment:

1. DMF server:

```
dmfserver# service dmfsoap stop      (if started)
dmfserver# service dmfman stop       (if started)
```

2. Parallel data-mover nodes:

```
mover# service dmf_mover stop
```

Note: Executing `service dmf_mover stop` on a mover node will cause existing data-mover processes to exit after the LS notices this change, which may take up to two minutes. The existing data-mover processes may exit in the middle of recalling or migrating a file; this work will be reassigned to other data-mover processes.

---

3. DMF server:

```
dmfserver# service dmf stop
```

4. Parallel data-mover nodes:

```
mover# service openvault stop
```

5. DMF server:

```
dmfserver# service openvault stop    (if OpenVault)
dmfserver# service tmf stop          (if TMF)
dmfserver# service pcp-storage stop  (if started)
dmfserver# service pcp stop
```

Using Out-of-Library Tapes

The mechanism for handling tape volumes that are physically not in the tape library varies depending on the mounting service:

- "TMF and Out-of-Library Tapes" on page 185
- "OpenVault and Out-of-Library Tapes" on page 185

TMF and Out-of-Library Tapes

When TMF is the mounting service, the `msgd`(8) command displays requests to import volumes. With TMF, DMF has no knowledge of what tapes are out of the library.

OpenVault and Out-of-Library Tapes

When OpenVault is the mounting service and a recall request is made for a tape that is physically not in the library, DMF will first try to read from copies on other volumes before it requests that the out-of-library tape be imported. If the reads from all other copies fail (for example, because their volumes are locked with the `hlock` flag), then DMF will request that the tape be imported.

If the desired volume is in the process of being written to, DMF will wait before requesting that a tape be imported; this occurs whether or not the `FORWARD_RECALLS` configuration parameter is set to `ON` (see "`volumegroup` Object" on page 410). If one of the copies is on a tape that is physically out of the library, DMF will wait for the drives to become usable even if the `REDIRECT_TIME_DG_DOWN` parameter has been set (see "`drivegroup` Object Parameters" on page 397).

DMF will request that the tape be imported before waiting for a copy in a library that has been permanently disabled with `ov_library -D`; see the `ov_library`(8) man page.

When a tape must be imported, DMF will send an email and issue an alert; you can use the `dmoper`(1) command to display these requests.

Example scenarios using volume groups named `vgA`, `vgB`, and `vgC`:

- Suppose that:
  - `vgA` contains the primary copy but is not in the library
  - `vgB` contains a secondary copy
  - `FORWARD_RECALLS` is configured `OFF` (the default)
  - The OpenVault library is enabled

  In this case, DMF will first try to recall from `vgB`. If that fails (for example, if `vgB`'s hold flags do not permit its use), then DMF will ask that the tape belonging to `vgA` be imported.
- Suppose that:

  - vgA contains the primary copy but its library is permanently disabled via the ov_library -D command
  - vgB contains a secondary copy whose tape is not in the library
  - vgC holds a tertiary copy but its tape is currently being written to
  - FORWARD_RECALLS is configured ON for all three VGs

  In this case, DMF will wait for the tape in vgC to become available. If that recall fails, then DMF will ask that the tape in vgB be imported. If that recall also fails, then DMF will wait for the library containing vgA to be enabled.

With OpenVault, also note the following:

- Tapes that are not in the library are not eligible for migrations or merging. See the hextern flag in "dmvoladm Directives" on page 555 for more information.
- The tapes that contain volume serial numbers used for backup tasks (as specified by the DUMP_TAPES configuration parameter) may not be out of the library.

Customizing DMF

You can modify the default behavior of DMF as follows:

- "File Tagging" on page 186
- "Site-Defined Policies" on page 187
- "Site-Defined Client Port Assignment in a Secure Environment" on page 188

File Tagging

File tagging allows an arbitrary 32-bit integer to be associated with specific files so that they can be subsequently identified and acted upon. The specific values are chosen by the site; they have no meaning to DMF.

Non-root users may only set or change a tag value on files that they own, but the root user may do this on any files. The files may or may not have been previously migrated.

To set a tag, use the dmtag(1) command or the libdmfusr.so library. For example:

```
% dmtag -t 42 myfile
```

To view the tag set for a given file, use the dmtag or dmattr commands. For example:

```
% dmtag myfile
42  myfile
% dmattr -a sitetag myfile
42
```

You can test a file's tag in the when clause of the following configuration parameters by using the keyword sitetag:

```
AGE_WEIGHT
CACHE_AGE_WEIGHT
CACHE_SPACE_WEIGHT
SELECT_LOWER_VG
SELECT_MSP
SELECT_VG
SPACE_WEIGHT
```

For example:

```
SELECT_VG fasttape when sitetag = 42
```

You can also access it in site-defined policies, as described below.

For more information, see the dmtag(1) man page.

Site-Defined Policies

Site-defined policies allow you to do site-specific modifications by writing your own library of C++ functions that DMF will consult when making decisions about its operation. For example, you could write a policy that decides at migration time which volume group (VG) or MSP an individual file should be sent to, using selection criteria that are specific to your site.

Note: If you customize DMF, you should inform your users so that they can predict how the user commands will work with your policies in place. You can add error, warning, and informational messages for commands so that the user will understand why the behavior of the command differs from the default.

For information about the aspects of DMF that you can modify, see Appendix C, "Site-Defined Policy Subroutines and the `sitelib.so` Library" on page 735.

Site-Defined Client Port Assignment in a Secure Environment

If you have a secure environment and if you require more than 512 active connections between DMF clients and the DMF server, you can specify that DMF assign a specific range of TCP ports.

Caution: If the environment is not secure, do not use this feature.

When a user executes a remote DMF client command, the user-command mechanism initiates a trusted `setuid root dmusrcmd`(8) process on the client. This `dmusrcmd` process performs all of the access validation required to send the user request the DMF server. By default, the remote `dmusrcmd` process verifies that it connects to the original client `dmusrcmd` process via a reserved port number in the range 512-1023; these port numbers are only available to trusted (`root`) processes. If all of those ports are in use, the `dmusrcmd` process will block until one of the trusted ports is free.

If your environment is secure (and therefore assigning a port that is not reserved for a trusted process is acceptable) and you require more than 512 active ports, you can create a `/usr/lib/dmf/dmf_client_ports` file on every potential DMF server and every DMF client. The file on the potential DMF servers must contain every port on which a client is allowed to connect, and the file on each DMF client must contain the ports that client is allowed to access. The file must be owned by `root` and have an access mode of `0600`.

You can use a range to specify the permitted ports. The format of the file is one or more lines as follows:

start_port_number:*end_port_number*

The *start_port_number* value must be greater than or equal to 512 and must be less than or equal to *end_port_number*. The order of the lines in the file is significant in

that `dmusrcmd` will start attempting to assign port numbers with the first line and process the lines in the order they appear in the file.

For example:

- To assign ports in the range 5000–6000:

  ```
  5000:6000
  ```

- To first assign ports in the range 7000–8000 and then assign ports in the range 5000–6000 (and never assign ports in the range 6001–6999):

  ```
  7000:8000
  5000:6000
  ```

If the file exists but is empty, `dmusrcmd` will not attempt to bind to any particular port numbers, but will use the non-secure port that is assigned to it by the kernel.

If the file does not exist, the default behavior of only assigning ports 512-1023 will be enforced.

Importing Data From Other HSMs

DMF has utilities to assist with importing data from filesystems managed by other hierarchical storage management (HSM) packages into DMF, provided that the filesystems to be imported are accessible via FTP or as local or NFS-mounted filesystems. These tools are not distributed with the DMF product. They are for use only by qualified SGI Professional Services personnel who assist sites doing conversions. For more information, contact SGI Professional Services.

DMF Manager

This chapter discusses the following:

- "Accessing DMF Manager" on page 192
- "Getting Started with DMF Manager" on page 192
- "Running Observer Mode or `admin` Mode" on page 195
- "Getting More Information in DMF Manager" on page 198
- "Setting Panel Preferences" on page 201
- "Refreshing the View" on page 202
- "Managing Licenses and Data Capacity with DMF Manager" on page 203
- "Configuring DMF with DMF Manager" on page 210
- "Displaying DMF Configuration File Parameters" on page 219
- "Starting and Stopping DMF and the Mounting Service" on page 220
- "Discovering DMF Problems" on page 221
- "Filtering Alerts" on page 225
- "Seeing Relationships Among DMF Components" on page 227
- "Managing Volumes" on page 229
- "Managing Libraries" on page 232
- "Displaying DMF Manager Tasks" on page 233
- "Monitoring DMF Performance Statistics" on page 233
- "Displaying Node Status" on page 252
- "Restoring Filesystems in DMF Manager" on page 255
- "Disaster Recovery in DMF Manager for Basic DMF" on page 262
- "Viewing the DMF Queue" on page 263
- "LTFS in DMF Manager" on page 263

Accessing DMF Manager

To access DMF Manager, do the following:

1. Point your browser to the following secure address:

   `https://`*YOUR_DMF_SERVER*`:11109`

2. Accept the security certificate.

   Note: DMF Manager generates its own SSL certificates, rather than having the SSL certificates signed by a commercial certificate authority. Therefore, the certificate warning is safe to ignore.

Also see "Running Observer Mode or `admin` Mode" on page 195.

Getting Started with DMF Manager

DMF Manager lets you configure DMF, install licenses, view the current state of your DMF system, and make operational changes.

When you initially open DMF Manager, you will see the **Overview** panel, which displays a high-level graphical view of the DMF environment and status for each DMF component, as shown in Figure 6-1. You can also configure DMF from this panel.

Each menu bar selection provides access to a DMF Manager panel, described in Table 6-1. To open a panel, click on the panel name in the menu. Right-click on the tab title to see its menu. Each panel has a key for its symbols.

Note: Some DMF Manager windows do not automatically update; choose the **Refresh** menu item to update an existing view.

Table 6-1 DMF Manager Panel Menus

Menu Bar Item	Panel Name	Description
Configuration	**Overview**	High-level graphical view of the DMF environment, status for each DMF component, and configuration capability
	Parameters	Details about the current parameter settings in the DMF configuration file and status for each DMF component
	Licenses	Information about installed DMF, CXFS, and SGI Management Center licenses and the ability to add and delete licenses (when logged in as `admin`)
Storage	**Volumes**	Status of *volumes*: physical tapes, SGI 400 virtual tape library (VTL) virtual tapes, COPAN massive array of idle disks (MAID) volumes, LTFS-assigned tapes, and unassigned tapes
	Libraries	Status of libraries, DMF drives, and LTFS drives; the ability to manage drives and their assignment to LTFS or DMF
Messages	**Reports**	Daily activity reports
	Alerts	Informational messages, warnings, and critical errors
	DMF Manager Tasks	Status of commands issued via DMF Manager that may take time to complete
Statistics	**DMF Activity**	Current and historical reports about the state and the performance of the DMF requests and throughput
	DMF Resources	Current and historical reports about the state and the performance of the DMF filesystems and hardware
	DMF I/O	Statistics about how DMF is using data and various kinds of specific media
QView	**QView**	Reports that provide details about the DMF processing queue and the ability to cancel or change the priority of a given request
Backups	**Recovery**	Capability to restore components from a list of available integrated-backup dumps and to restore filesystems from any DMF backup session
Help	**Getting Started**	This section

Menu Bar Item	Panel Name	Description
	Admin Guide	This manual
	About DMF Manager	Version and copyright information about the tool

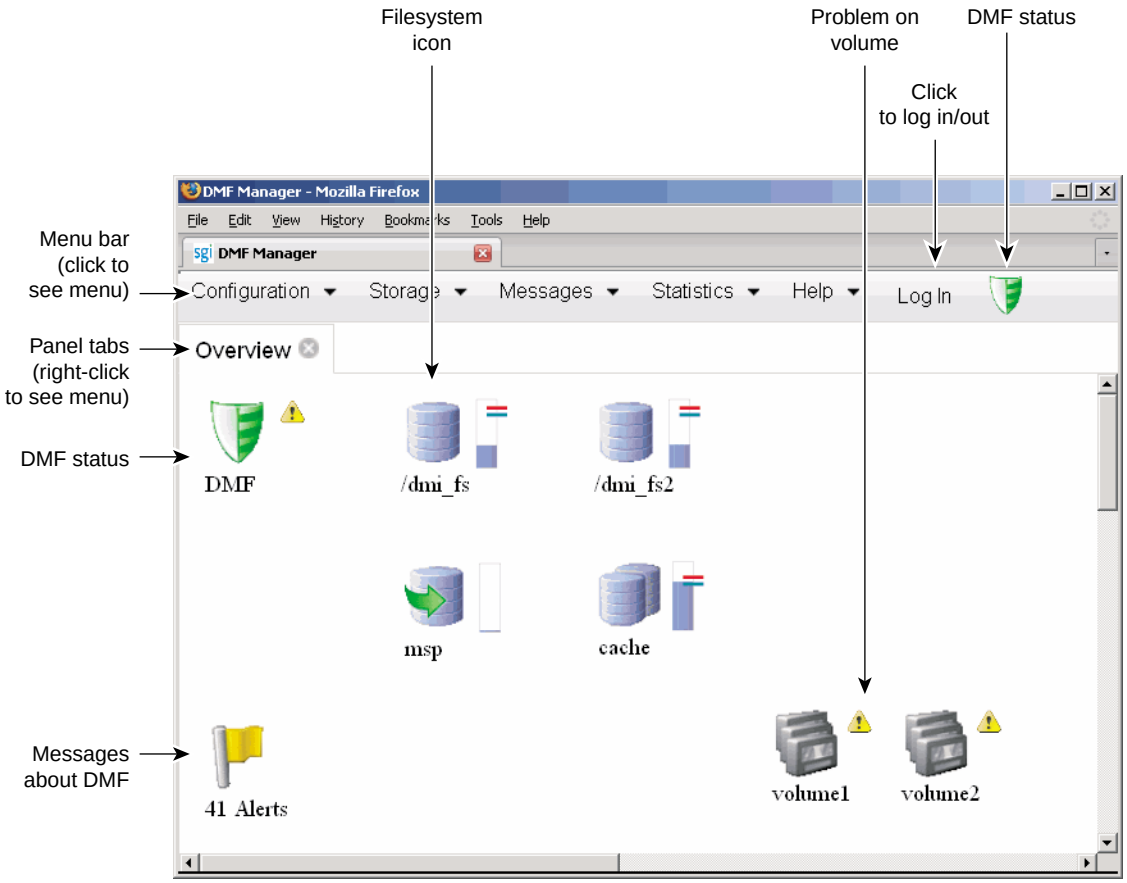

Figure 6-1 DMF Manager Overview Panel

Some panels have expandable folders. Click on the "+" symbol to expand a folder or on the "—" symbol to contract it, or use the **Expand All** and **Collapse All** buttons. Click on a report name to display the associated graphs. For example, see Figure 6-21 on page 229.

Running Observer Mode or `admin` Mode

You can run DMF Manager in observer mode (the default) or you can log in to `admin` mode for more functionality, as described in the following sections:

- "Observer Mode Functionality" on page 195
- "`admin` Mode Functionality" on page 196
- "Admin Mode Access" on page 198

Observer Mode Functionality

In the default observer mode in DMF Manager, you can do the following:

- View the state of DMF and the mounting service. See:
  - "Getting Started with DMF Manager" on page 192
  - "Discovering DMF Problems" on page 221
- View the fullness of each managed filesystem. See:
  - "Getting Started with DMF Manager" on page 192
  - "Discovering DMF Problems" on page 221
- View the licensed capacity. See "Showing Current DMF Usage and Licensed Capacity" on page 205.
- View installed DMF, CXFS, XVM, and SGI Management Center licenses and the system information required to request a new license. See "Managing Licenses and Data Capacity with DMF Manager" on page 203.
- View DMF's configuration. See "Displaying DMF Configuration File Parameters" on page 219.
- View relationships among DMF components. See "Seeing Relationships Among DMF Components" on page 227.
- Get context-sensitive help and view the DMF administration guide. See "Getting More Information in DMF Manager" on page 198.

- View information about volumes:
  - View each volume's status and fullness
  - Sort which volumes to view
  - Display dump tapes (physical tapes and SGI 400 VTL virtual tapes)
  - View the status of each drive
  - Temporarily create a user-defined query

  Note: Saving the query requires admin mode. See "`admin` Mode Functionality" on page 196.

  See:
  - "Getting Started with DMF Manager" on page 192
  - "Managing Volumes" on page 229
- View the status of libraries, LTFS tapes, and unassigned tapes that could be assigned to LTFS
- View the daily reports (with history) and DMF alerts. See "Discovering DMF Problems" on page 221.
- View the long-running DMF Manager tasks (those currently executing and a history of executed tasks). See "Displaying DMF Manager Tasks" on page 233.
- View current and historical reports about DMF activity and resources. See "Monitoring DMF Performance Statistics" on page 233.

`admin` Mode Functionality

If you log in to `admin` mode, you can perform the additional tasks such as the following:

- Add licenses to or delete licenses from the `/etc/lk/keys.dat` system license file or the DMF license file specified by the `LICENSE_FILE` configuration parameter (see "`base` Object Parameters" on page 278).
- Start/stop DMF and the mounting service. See "Starting and Stopping DMF and the Mounting Service" on page 220.

- Create or modify the DMF configuration. See "Configuring DMF with DMF Manager" on page 210.
- Create and modify the LTFS configuration. See "Configuring LTFS using DMF Manager" on page 265.
- Mount, unmount, and assign LTFS tapes; reserve a drive for LTFS use or reassign a drive to DMF. See "LTFS in DMF Manager" on page 263.
- Manage volumes (physical tapes, SGI 400 VTL virtual tapes, and COPAN MAID volumes):
  - Change the hold flags
  - Manually mark a volume as *sparse*, meaning containing blank or inactive areas after data has been deleted. (The data from a sparse volume will be later moved to another volume via *volume merging*, the mechanism provided by the LS for copying active data from volumes that contain largely obsolete data to volumes that contain mostly active data.) For more information, see "Volume Merging" on page 543.

    Note: Merging is not appropriate for a volume configured as a fast-mount cache.

  - Empty a damaged volume of all useful data and restore another copy in the volume group (VG)
  - Eject physical tape cartridges from the tape library
  - Load physical tape cartridges into the tape library and configure them for DMF's use with OpenVault
  - Read data to verify the volume's integrity
  - Enable / disable drives

  See "Managing Volumes" on page 229.
- Acknowledge DMF alerts. See "Discovering DMF Problems" on page 221.
- Control long-running DMF Managed tasks (acknowledge, suspend / resume, or kill). See "Displaying DMF Manager Tasks" on page 233.

Admin Mode Access

To log in to DMF Manager as the admin user, click the **Log In** button in the upper-right corner of the window, as shown in Figure 6-1 on page 194.

The default `admin` password is `INSECURE`. You should change the `admin` password and only provide it to those persons who you want to make administrative changes. (After you change the `admin` password, you cannot administratively set it to `INSECURE` again.)

Note: If you are upgrading from a release prior to DMF 5.4, the `admin` password will be reset to `INSECURE`. You should change the `admin` password to a site-specific value after upgrading.

Getting More Information in DMF Manager

Each panel that uses icons has a key for its symbols, available via the **Show Key** menu selection. Figure 6-2 shows the key to icons on the **Overview** panel.

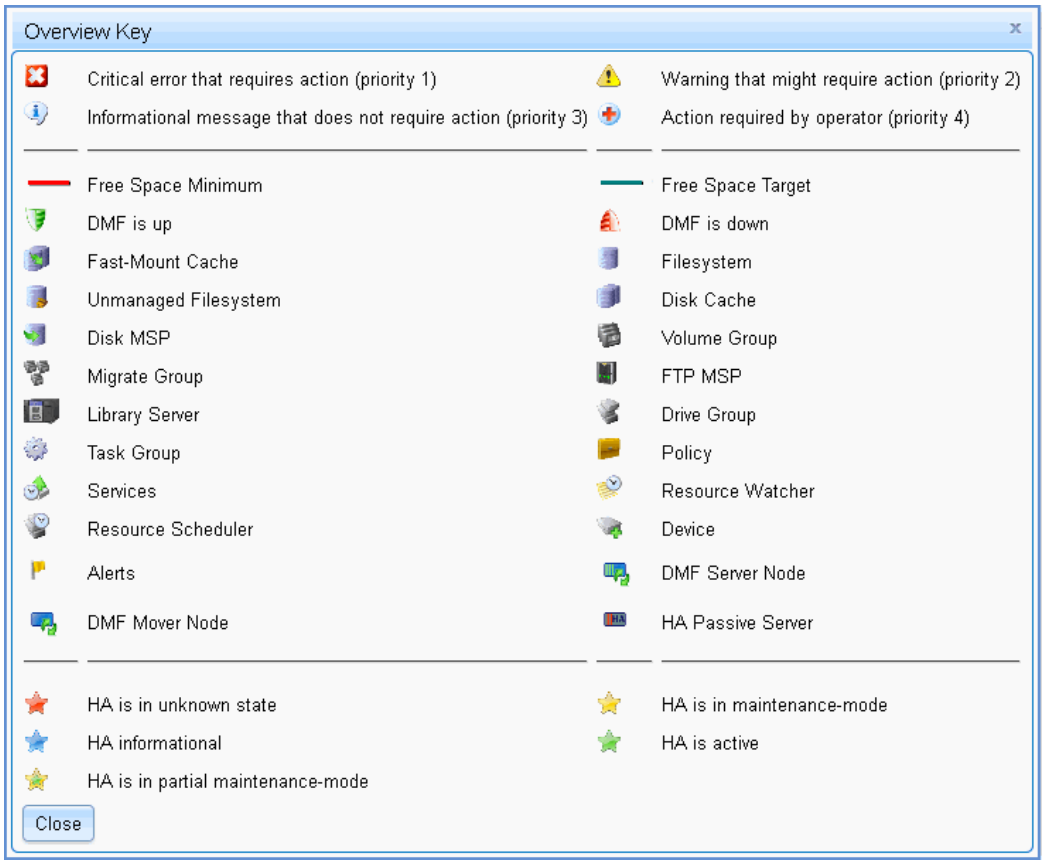

Figure 6-2 Overview Key to Symbols

To display information about an object, you can move the mouse button over the object, as shown for the `burn` server in Figure 6-3.

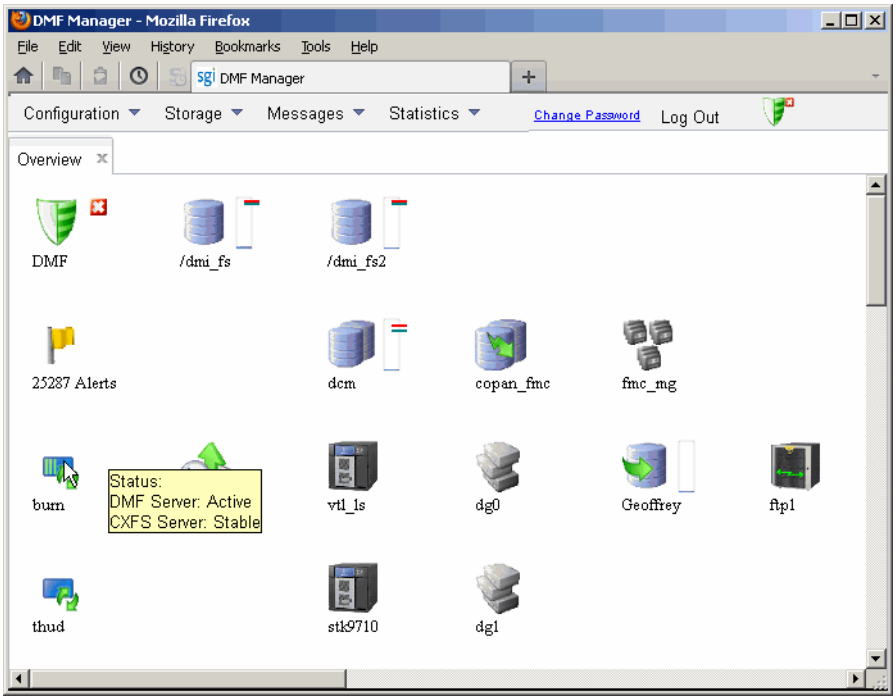

Figure 6-3 Displaying Information About an Icon

To get more information about any item, right-click on it and select the **What is this?** option. For example, Figure 6-4 shows the help text for the **Alerts** icon.

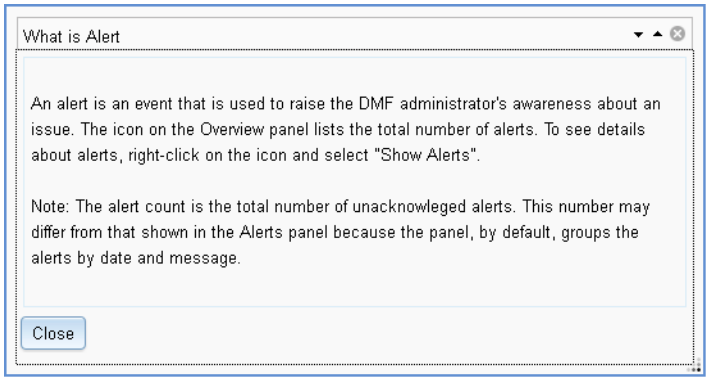

Figure 6-4 "What Is ..." Information

Each panel also has a **What is '*PanelName*'?** menu selection.

For a quick-start to using DMF Manager, select the following from the menu bar:

Help
> Getting Started

To access the DMF administrator guide (this manual), select the following:

Help
> Admin Guide

Setting Panel Preferences

Each DMF Manager panel (other than the **Help** panels) has a **Set *PanelName* Preferences** menu item that allows you to vary what is shown on the panel, how it behaves, and how often it is refreshed (see "Refreshing the View" on page 202).

For example, Figure 6-5 shows the preferences that you can set for the **Overview** panel.

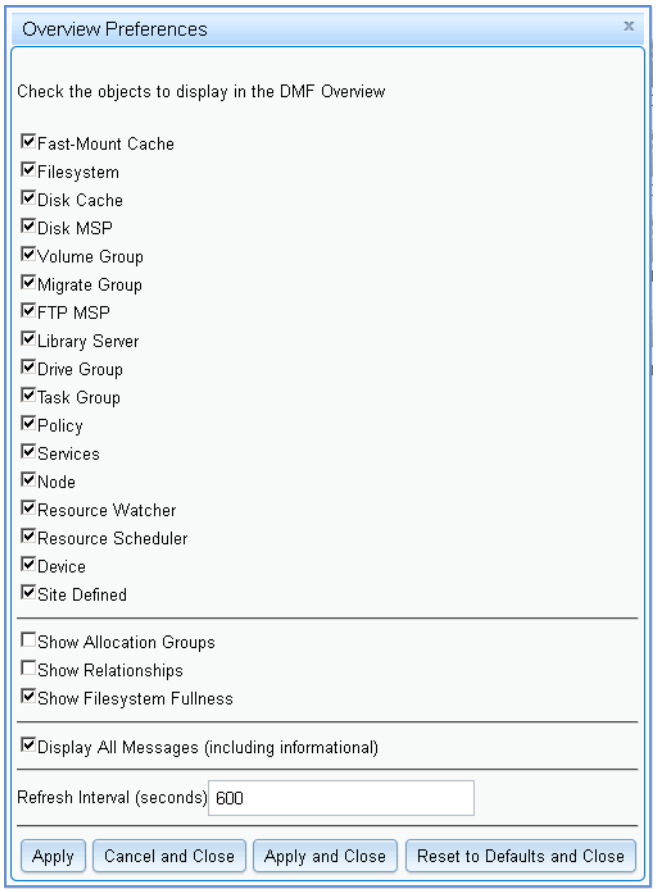

Figure 6-5 DMF Manager Overview Preferences Panel

Refreshing the View

Some DMF Manager panels refresh automatically by default but others do not. To refresh a panel, choose the **Refresh *PanelName*** menu item.

Caution: If you refresh the **Overview** panel while in configuration mode, any changes that have been made but not saved or applied will be lost and you will exit from configuration mode.

You can use **Set *PanelName* Preferences** menu to set an automatic refresh interval for individual panels. See "Setting Panel Preferences" on page 201.

Note: A refresh interval that is too short can cause contention between the DMF server and the browser. On heavily used systems, some displays may not be refreshed at extremely low intervals because the time to gather the information exceeds the refresh time. In such cases, you will only see a refresh as often as one can be completed.

Managing Licenses and Data Capacity with DMF Manager

This section discusses the following:

- "Adding New Licenses" on page 203
- "Deleting Existing Licenses" on page 204
- "Viewing the Installed Licenses" on page 205
- "Showing Current DMF Usage and Licensed Capacity" on page 205
- "Showing Remaining Storage Capacity" on page 206

For more information, see Chapter 2, "DMF Licensing" on page 67.

Adding New Licenses

To add one or more a new licenses, do the following:

1. Gather the required host information by viewing the **Licenses** panel. For more information, see "Gathering the Host Information" on page 74.
2. Obtain the required keys from SGI. See "Obtaining the License Keys" on page 75.
3. Log in to DMF Manager as `admin`.
4. Paste the keys into the text-entry area of the **Licenses** panel, highlighted in Figure 6-6.
5. Click the **Add license** button.

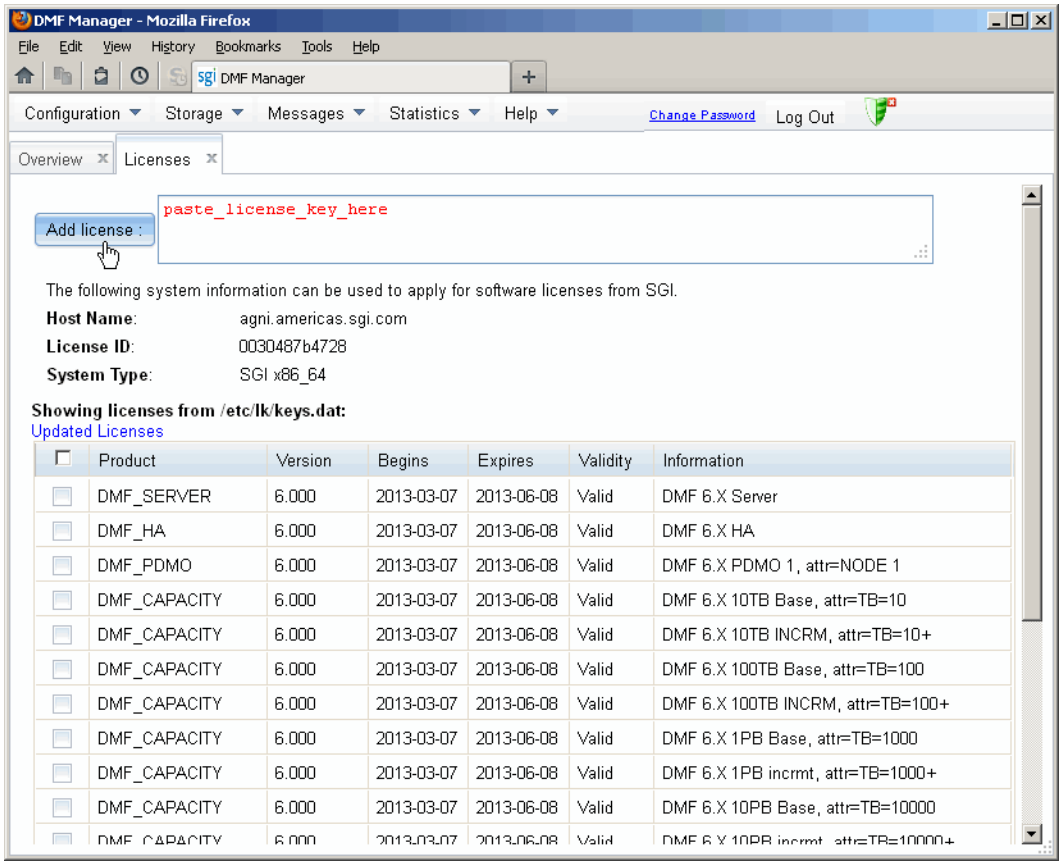

Figure 6-6 Adding a License Key in DMF Manager

In a DMF server HA configuration, you can add licenses to all potential DMF servers. Select the node to be acted on by choosing its name from the **Showing license from /etc/lk/keys.dat on** prompt.

Deleting Existing Licenses

To delete one or more existing licenses, do the following:

1. Log in to DMF Manager as `admin`.

2. Select the licenses you want to delete by clicking their check boxes.
3. Right-click anywhere in the table and select **Delete selected licenses**.

In an HA configuration, you can delete licenses from all potential DMF servers in the HA cluster. Select the node to be acted on by choosing its name from the **Showing license from /etc/lk/keys.dat on** prompt.

Viewing the Installed Licenses

To see the currently installed licenses for the server running DMF Manager, select:

Configuration
> Licenses

The **Licenses** panel lists the currently installed DMF, CXFS, XVM, and SGI Management Center licenses. By default, the licenses display in the same order in which they appear in the `/etc/lk/keys.dat` file. You can sort the table by clicking on the desired column header. To resize a column, select the boundary divider in the table header.

In an HA configuration, you can view licenses from all potential DMF servers in the HA cluster. Select the node to be acted on by choosing its name from the **Showing license from /etc/lk/keys.dat on** prompt.

Showing Current DMF Usage and Licensed Capacity

To determine the current DMF usage and licensed capacity, right-click on the DMF icon in the **Overview** panel and select **Show Usage...**.

You can also display this information by selecting the following:

Configuration
> Licenses
> Show Usage...

Figure 6-7 shows an example. For more information, see "Displaying Current DMF Data Capacity Use" on page 72.

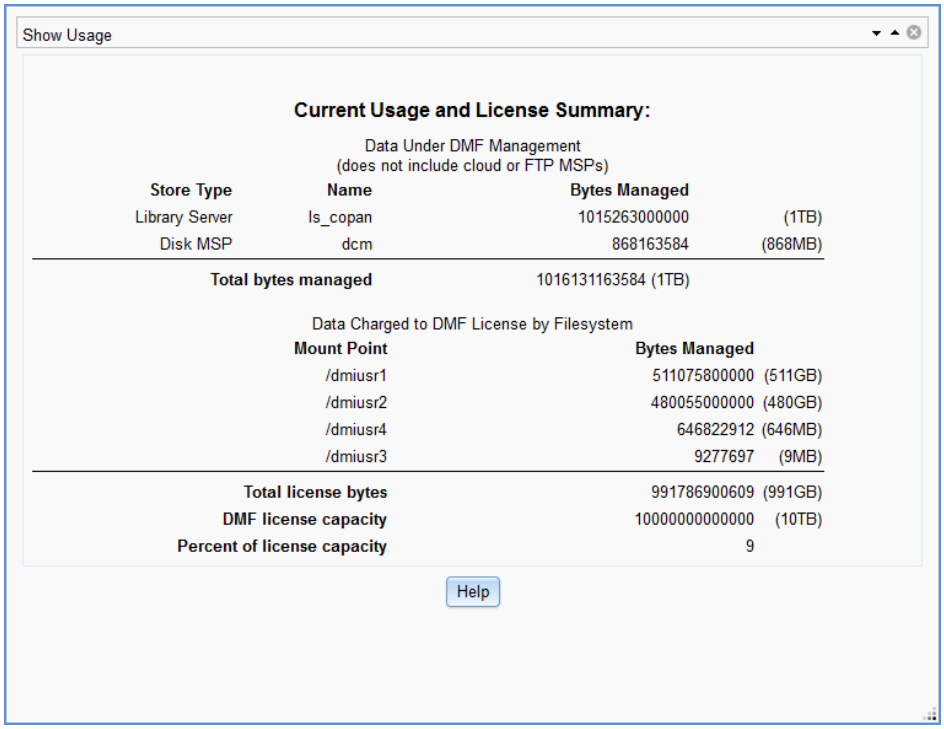

Figure 6-7 Data Under Management and Data Charged to the Licenses

Showing Remaining Storage Capacity

To display the total capacity, an estimate of the current total migrated data that is active, and an estimate of the writable space that is currently available, right-click on the DMF icon in the **Overview** panel and select **Show Capacity**. For example, Figure 6-8 shows the reports for two library servers.

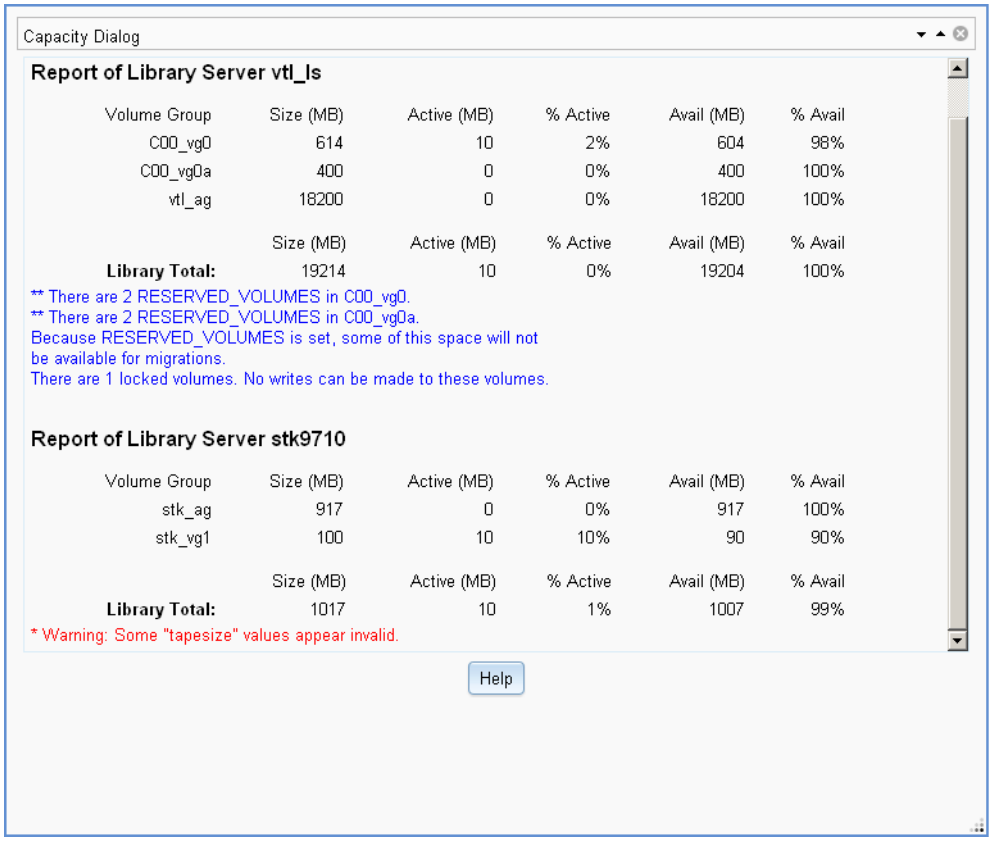

Figure 6-8 DMF Capacity

Note: To see accurate estimates, you must first set the size of each volume accurately. For details, see the `dmvoladm`(8) man page.

This displays an estimate of the remaining storage capacity for each volume group in each LS. It reports the following totals for all volumes in the listed VGs and LSs (data compression of data is not taken into account):

Field	Description
Volume Group	The name of volume group
Size	The total capacity in megabytes (MB)
Active	The total migrated data (in MB) that may be recalled (also represented as a percentage)
Avail	The total writable space (in MB) on all volumes within the VG or LS (also represented as a percentage)

Locked volumes are noted and an informational message highlights their number for each LS.

For example, Figure 6-9 shows that the `C00_vg0` and `C00_vg0` VGs are almost full and therefore there might not be any more space available for migrations because they each have 2 volumes reserved for merges (set by the `RESERVED_VOLUMES` parameter, see "`volumegroup` Object" on page 410).

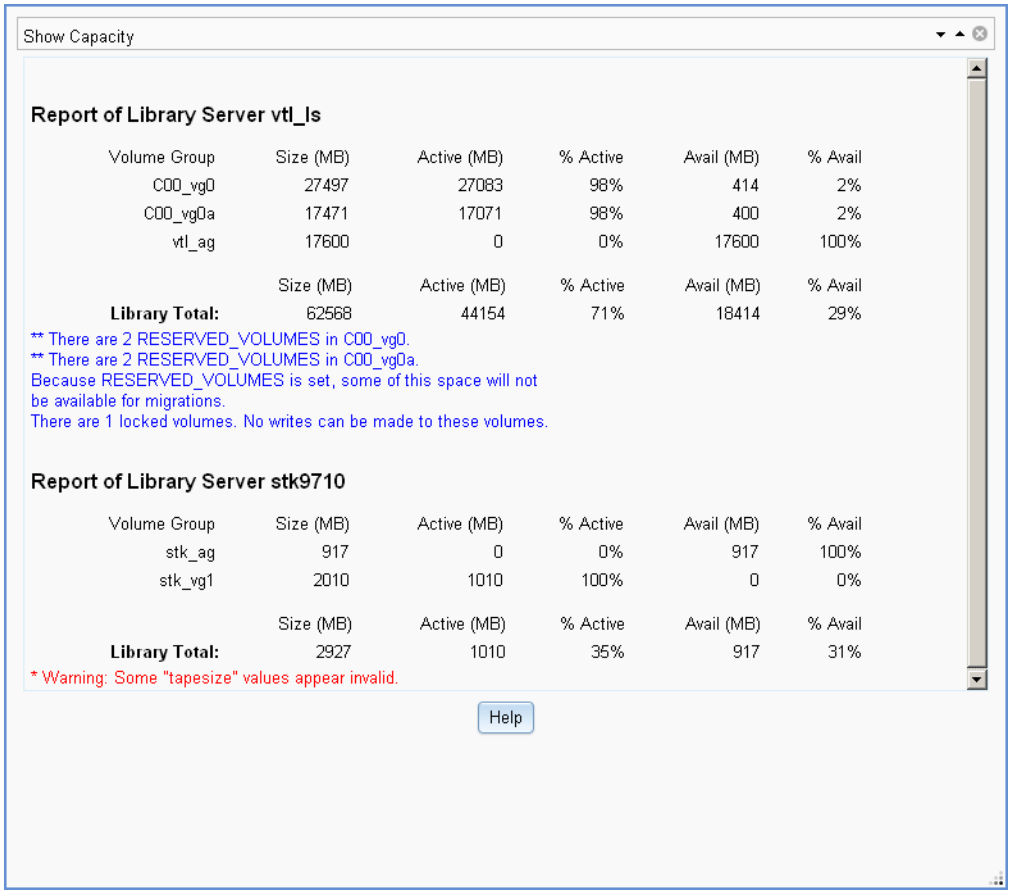

Figure 6-9 Remaining DMF Capacity

To display more information, such as volume size, select:

Volumes
> Set Volumes Preferences

To see details about specific volumes, select:

Storage
> Volumes

Check the desired items to display.

There may be additional space that can be reclaimed from data that was written and has since been hard deleted. For more information, see "Volume Merging" on page 543.

For more information about how the calculations are made, see the `dmcapacity`(8) man page.

Configuring DMF with DMF Manager

You can establish and edit the DMF configuration by logging in as the admin user and using the **Overview** panel. If you make a change to the configuration, the background color will change to gray wallpaper displaying “Configuration mode”, indicating that you must save or cancel your changes.

This section discusses the following:

- "Limitations to the DMF Configuration Capability" on page 211
- "Showing All Configured DMF Objects" on page 211
- "Setting Up a New DMF Configuration File" on page 212
- "Copying an Object" on page 215
- "Modifying an Object" on page 217
- "Creating a New Object" on page 217
- "Deleting an Object" on page 218
- "Validating Your Changes" on page 218
- "Saving Your Configuration Changes" on page 218
- "Exiting the Temporary Configuration without Saving" on page 219

Limitations to the DMF Configuration Capability

The configuration capability in DMF Manager has the following limitations:

- Comments are not permitted in the configuration file created or modified by DMF Manager. If you edit an existing configuration file that has comments and save the file, the comments will be deleted from the updated configuration file.

  Note: The original DMF configuration file, including the comments, will be preserved in a time-stamped copy (`/etc/dmf/dmf.conf.`*TIMESTAMP*).

- **Adding** site-specific objects or site-specific parameters is not supported (if site-specific items already exist in the DMF configuration file, they are preserved).
- DMF Manager cannot detect if multiple users have logged in as `admin` and are therefore capable of overwriting each other's changes. At any given time, only one user should log in as `admin` and make configuration changes.

Showing All Configured DMF Objects

To see all currently configured objects, select:

Overview
> Configure...
> Show All Configured Objects

By default, all currently configured objects will also be shown after you make a configuration change and select **Continue**.

After you either save or cancel the configuration changes, the icons that are displayed will return to the preferences you have set. See "Setting Panel Preferences" on page 201.

Setting Up a New DMF Configuration File

Note: For preliminary required steps, including installing the required software and creating the required filesystems, see "Overview of the Installation and Configuration Steps" on page 165.

When you first access DMF Manager, you must enter the DMF Manager admin password, which brings you to the sample configuration page that contains templates for the `base` object and `taskgroup` objects for daemon tasks and migration tasks. To create an appropriate DMF configuration file, individually add the required objects or select one of the preconfigured samples for assistance:

Overview
> Configure ...
> Pre-Configured
> *sample_name*

You can also access this menu by right-clicking anywhere in the **Overview** panel.

The preconfigured items provide a starting point of objects that you can modify with specific information for your site. For example, Figure 6-10 shows the icons that will appear after you select **DCM MSP Sample**. The gray wallpaper indicates that the sample file has been loaded. The errors displayed will disappear after you validate the configuration.

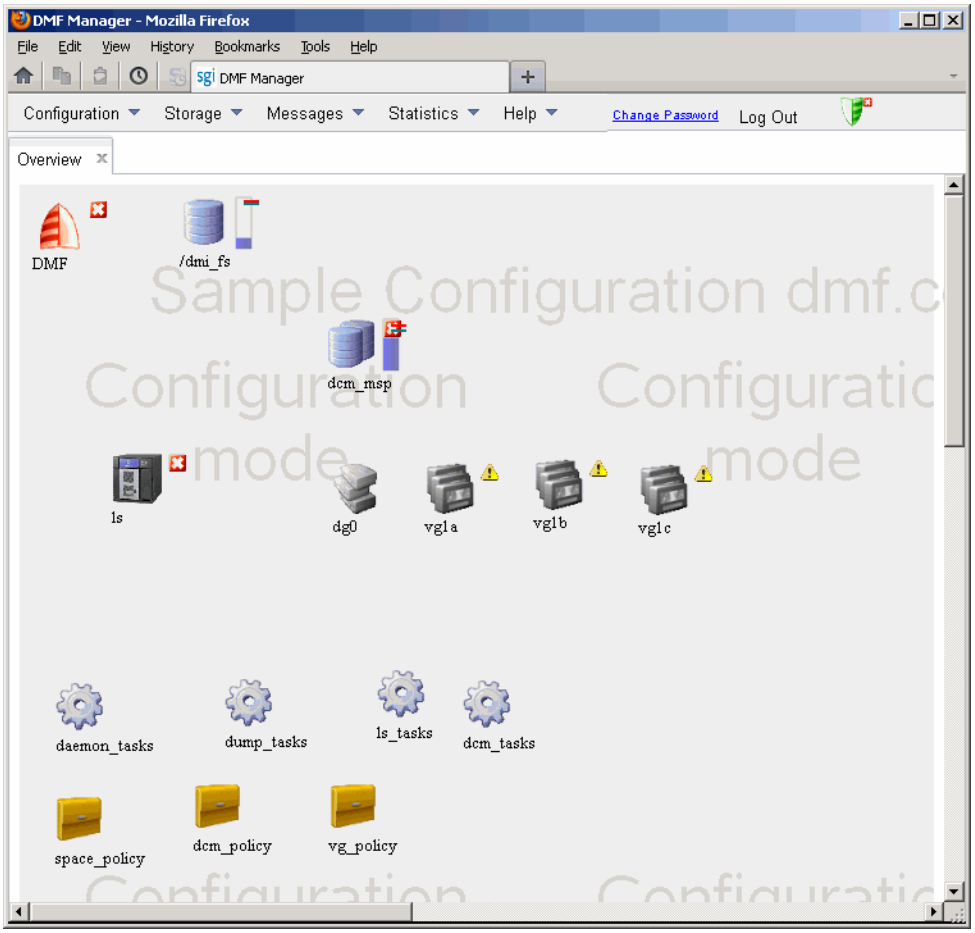

Figure 6-10 Temporary Workspace for a Preconfigured DCM MSP Sample

For example, the **DCM MSP Sample** selection includes the following objects:

- `base` object and `dmdaemon` object (represented by the DMF shield icon):
- `filesystem` object named `/dmi_fs`
- `msp` object named `dcm_msp` configured for a DCM MSP
- `libraryserver` object name `ls`

- `drivegroup` object name `dg0`
- `volumegroup` objects named `vg1a`, `vg1b`, `vg1c`
- `taskgroup` objects named `daemon_tasks`, `dump_tasks`, `ls_tasks`, and `dcm_tasks`
- `policy` objects named `space_policy`, `dcm_policy`, and `vg_policy`configured for automated space management and MSP selection

For more information about these objects and their parameters, see Chapter 7, "DMF Configuration File" on page 271.

You can then modify the sample configuration as needed. See:

- "Copying an Object" on page 215
- "Modifying an Object" on page 217
- "Creating a New Object" on page 217
- "Deleting an Object" on page 218
- "Validating Your Changes" on page 218
- "Saving Your Configuration Changes" on page 218

To exit a preconfigured sample without saving any of your changes, select:

Overview
> Configure...
> Cancel Configuration

Copying an Object

To copy an object, right-click on it and select:

Configure ...
> Copy

Then name the new object and enter the values you desire for the object's parameters. For example, Figure 6-11 shows naming a copied filesystem object `/dmi_fs2`.

Note: Many parameters have default values, but these are not necessarily shown in the DMF Manager windows. Only explicitly specified values are shown by DMF Manager. If a parameter has no value specified, its default value is assumed.

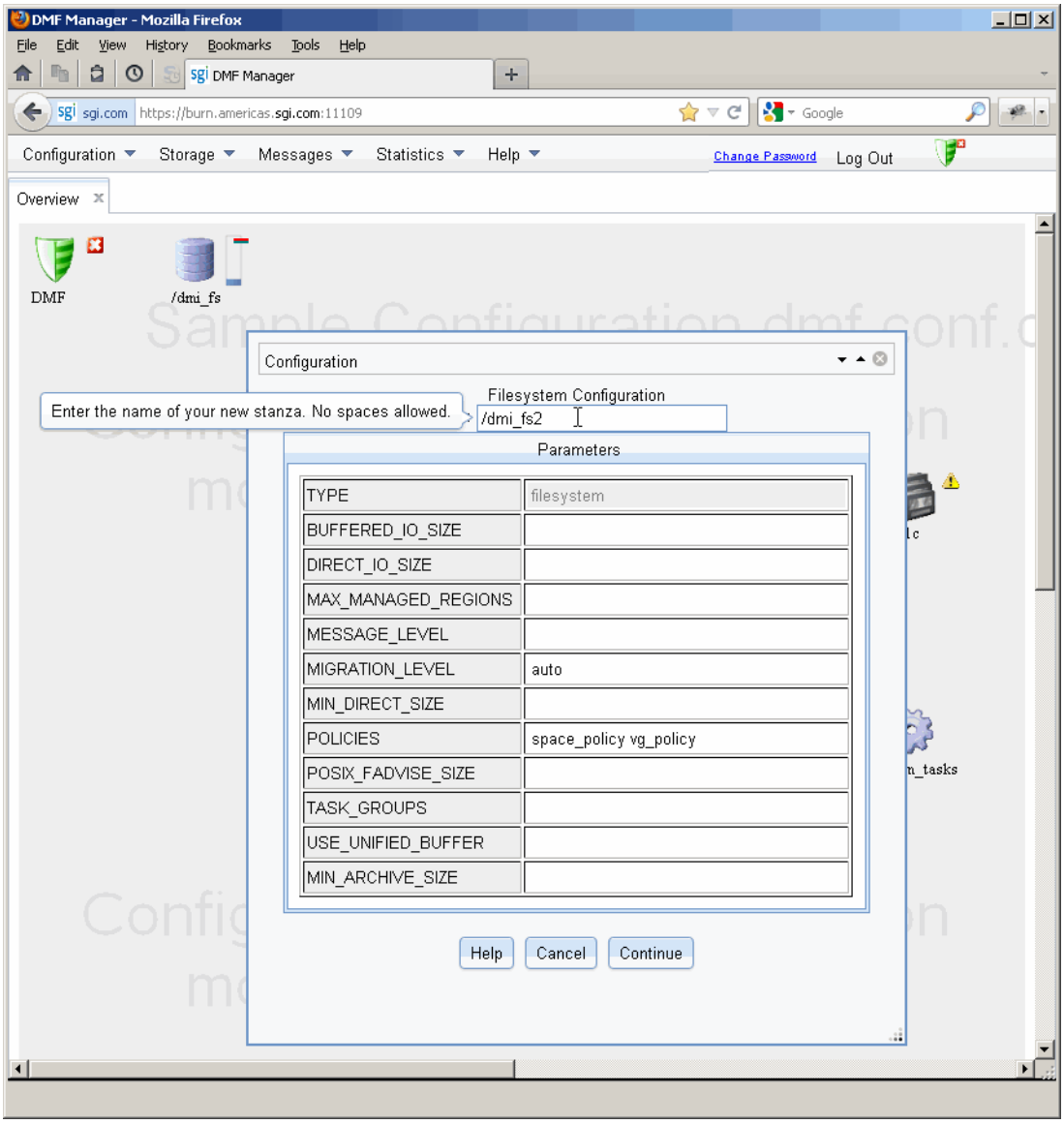

Figure 6-11 Naming a Copied Object

To get more information about a parameter, right-click on it and select the **What is this?** option. See "Getting More Information in DMF Manager" on page 198.

To make your changes appear in the **Overview** display, select **Continue**. To permanently save your changes, see "Saving Your Configuration Changes" on page 218.

Modifying an Object

To edit the parameters for an existing object, right-click on it and select:

Configure ...
> Modify

Then enter the values you desire for the parameters shown. To get more information about a parameter, right-click on it and select the **What is this?** option. See "Getting More Information in DMF Manager" on page 198.

To rename an object, delete it and create a new object. See:

- "Copying an Object" on page 215
- "Creating a New Object" on page 217
- "Deleting an Object" on page 218

To make your changes in the temporary configuration view, select **Continue**. To permanently save your changes, see "Saving Your Configuration Changes" on page 218.

Creating a New Object

To create a new object, right-click on blank space anywhere in the **Overview** panel and select the object. Also see "Setting Up a New DMF Configuration File" on page 212.

You can also right-click on an existing object and create another empty object of the same type by selecting:

Configure ...
> Add New

Then name the object and enter the values you desire for the parameters shown. To get more information about a parameter, right-click on it and select the **What is this?** option. See "Getting More Information in DMF Manager" on page 198.

To make your changes appear in the **Overview** display, select **Continue**. To permanently save your changes, see "Saving Your Configuration Changes" on page 218. Also see "Exiting the Temporary Configuration without Saving" on page 219.

Deleting an Object

To delete an object, right-click on it and select:

Configure ...
> Delete

Validating Your Changes

To verify that your changes to the temporary configuration are valid, select the following:

Overview
> Configure ...
> Validate Configuration

Saving Your Configuration Changes

To make your changes appear in the **Overview** display for this DMF Manager session, click **Continue** after creating or modifying an object. (This does not change the DMF configuration file.)

To save the temporary configuration so that you can work on it later, select:

Overview
> Configure ...
> Save Temporary Configuration

To permanently save your changes and apply them to the DMF configuration file, do the following:

1. Verify that your changes are valid. See "Validating Your Changes" on page 218.

2. Select:

   Overview
   > Configure ...
   > Apply Configuration

Exiting the Temporary Configuration without Saving

To exit the temporary configuration entirely without saving any of your changes, select:

Overview
> Configure...
> Cancel Configuration

The **Configure** menu is also available by right-clicking within the **Overview** display. If you refresh the screen, the temporary configuration will also be canceled.

Displaying DMF Configuration File Parameters

The following menu bar selection displays the contents of the DMF configuration file:

Configuration
> Parameters

For example, Figure 6-12 shows the configuration parameters for a drive group. For information about any individual parameter, right-click on it and select the **What is** option.

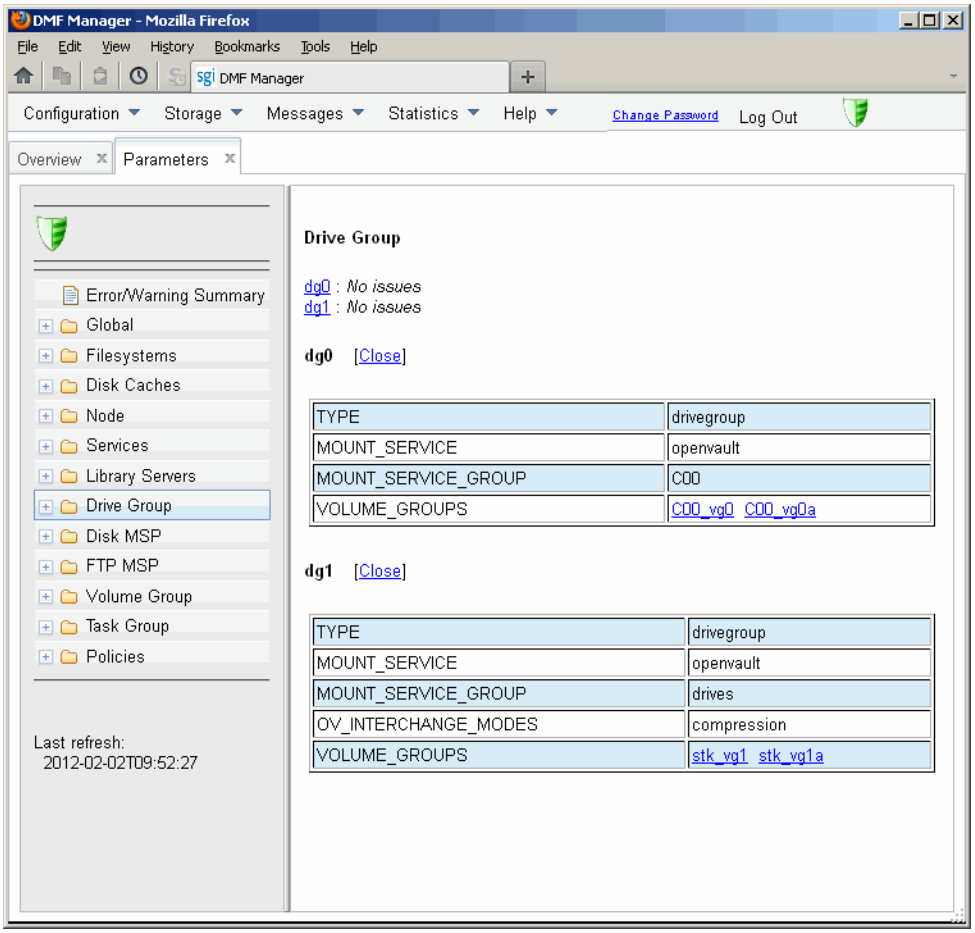

Figure 6-12 DMF Configuration Parameters in DMF Manager

Starting and Stopping DMF and the Mounting Service

To start or stop DMF and the mounting service, do the following:

1. Log in as the `admin` user.
2. Right-click on the DMF icon in the **Overview** panel.
3. Select the desired action.

Discovering DMF Problems

DMF Manager denotes issues by adding a red or yellow icon next to the component that is experiencing problems. For example, Figure 6-13 shows that although DMF is still running, there is a potential problem. To investigate, hover the mouse over the shield icon to display pop-up help that details the warning.

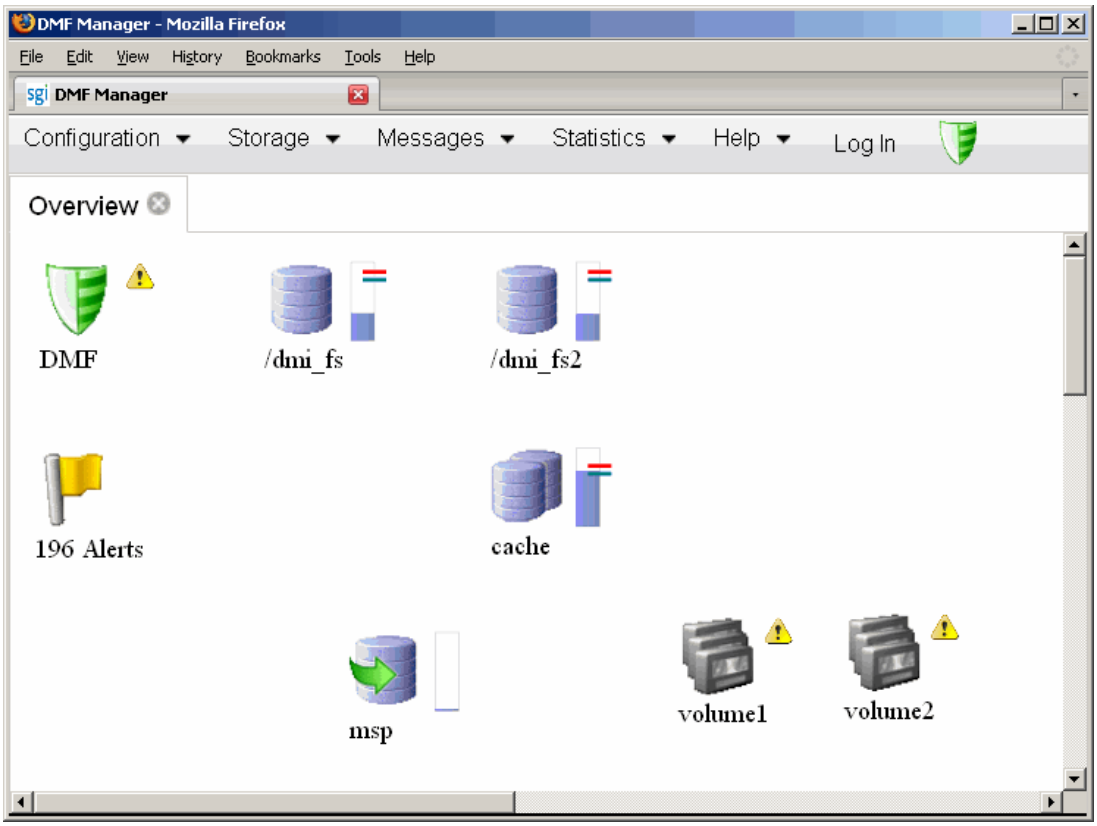

Figure 6-13 DMF Manager Showing Problems in the DMF System

For more information, right-click the **Alerts** icon flag and select **Show Alerts...** or choose the following from the menu bar:

Messages
> Alerts

Either action will open the **Alerts** panel, which displays the unacknowledged alerts (by default, grouped by date and message) with the following sortable fields:

- **Time** is the date and time at which a particular alert was issued (by default, alerts are sorted by time from most recent to oldest)
- **Alert Message** is the notice, warning, or critical error reported during the operation of DMF
- **Priority** is an icon as shown in Figure 6-14 that represents the severity of the alert
- **Host** is the node that issues the alert
- **Count** is the number of times this particular alert has been issued within one calendar day

---

Note: By default, identical alerts are grouped and only the time that the last alert was issued is displayed. To view all alerts and their corresponding time stamps, deselect the **Group identical alerts within a day** box in the **Alerts Preferences** panel.

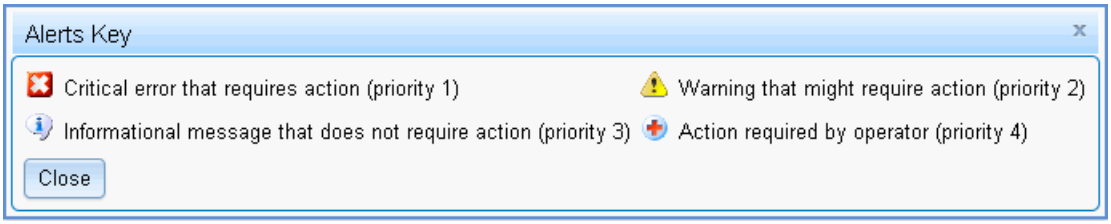

Figure 6-14 Alerts Key

Figure 6-15 shows an example of unfiltered alerts.

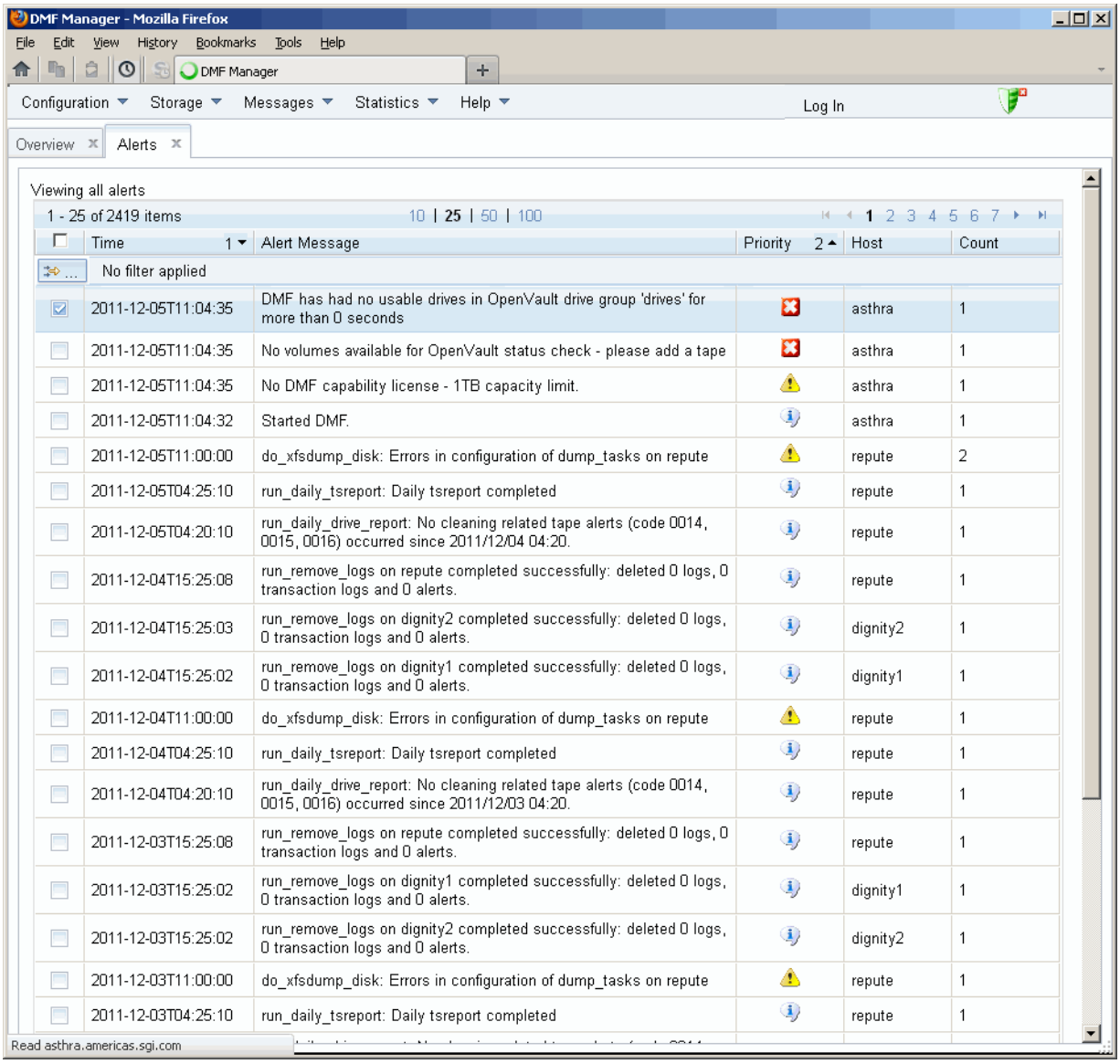

Figure 6-15 Unfiltered Alerts

For more information about an alert, select it and choose **Help on this alert**, such as shown in Figure 6-16. To customize the display, see "Filtering Alerts" on page 225.

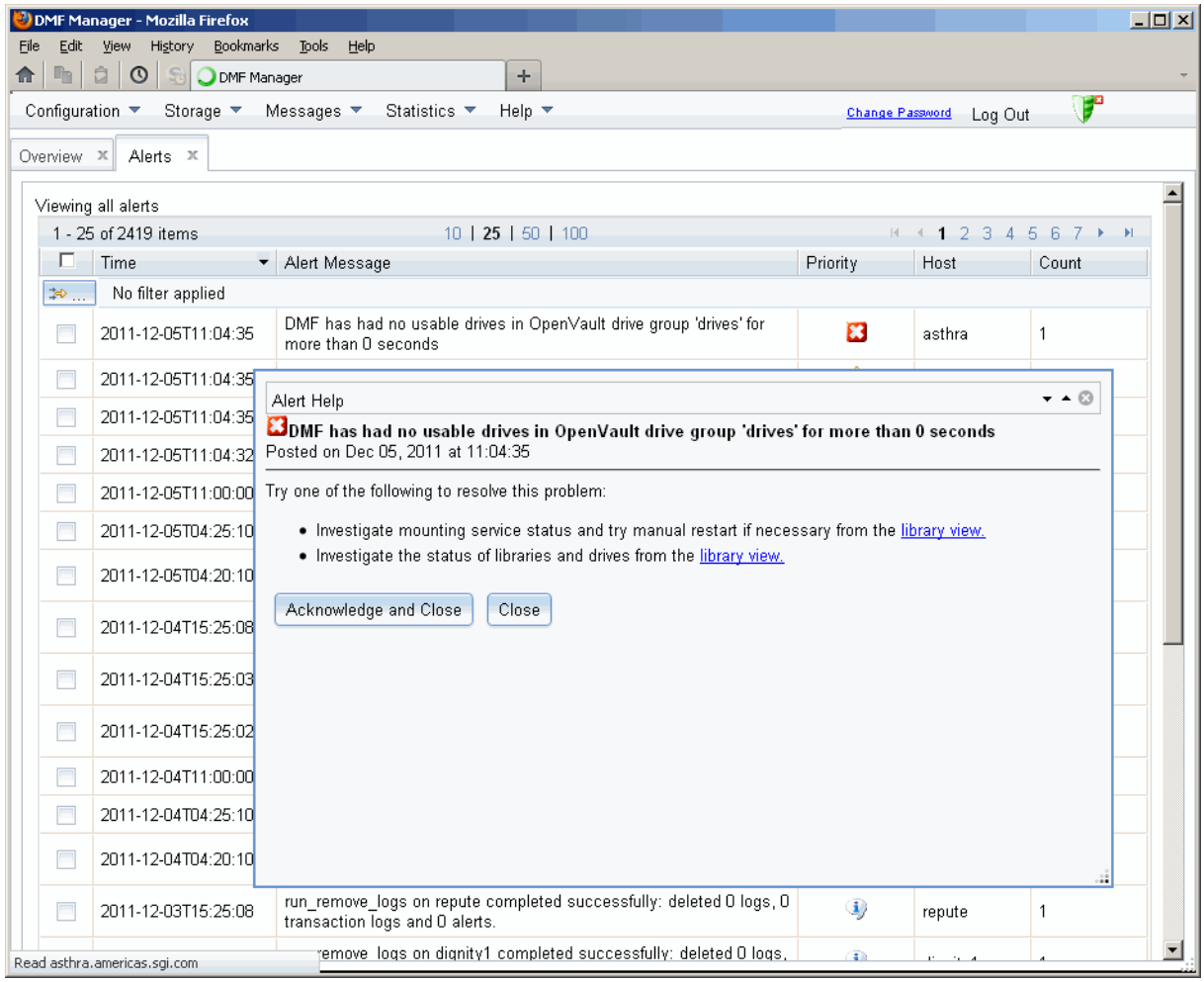

Figure 6-16 DMF Manager Alerts Panel and Help Information

If you are logged in, you can acknowledge selected alerts or clear all alerts. See "Running Observer Mode or `admin` Mode" on page 195.

You can also use the following panel to view daily activity reports (those containing critical log errors show red warning symbols):

Messages
> Reports

Also see "Viewing the DMF Queue" on page 263.

Filtering Alerts

You can customize the **Alerts** display by applying one or more filters.

For example, to show critical errors and warnings about OpenVault sent on December 5, you could establish three filters:

1. Click in the filter bar, as shown in Figure 6-17.

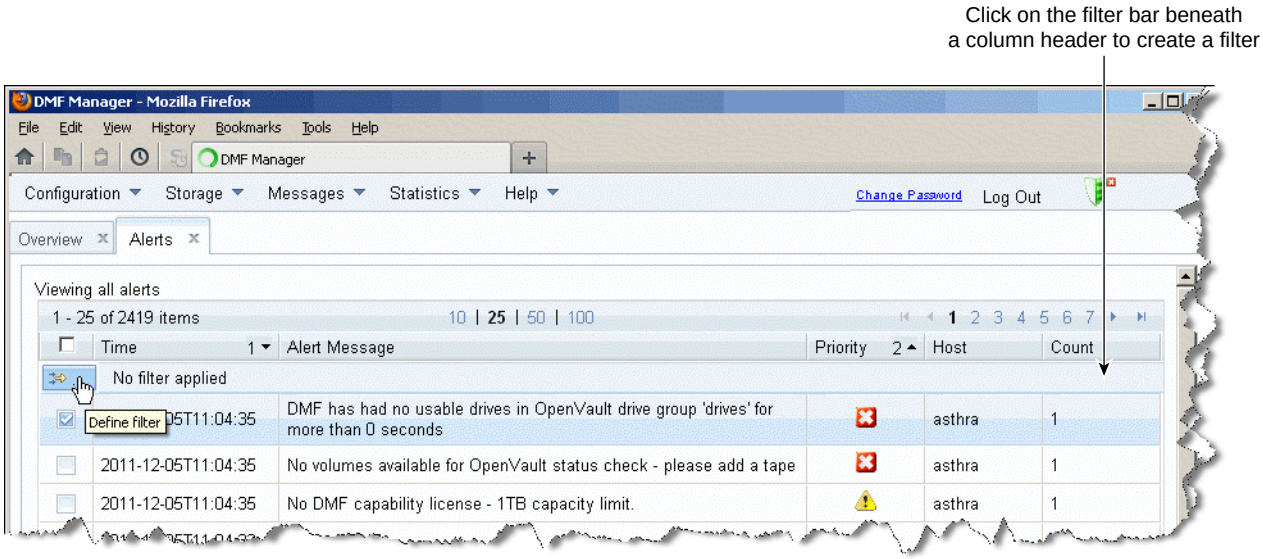

Figure 6-17 Define Filters for Alerts

2. **Rule 1**:

- For **Column**, select **Time**

Note: If you click in the filter bar below a column header, the column name will be selected automatically in the **Filter** dialog.

- For **Condition**, select **is**
- For **Value**, select December **5**

3. Click the green plus sign to add another rule, as shown in Figure 6-18.

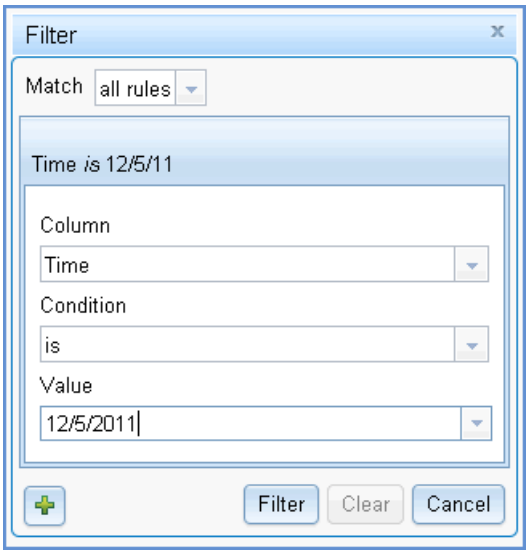

Figure 6-18 Adding Another Filter Rule

4. **Rule 2**:
   - For **Column**, select **Priority**
   - For **Condition**, select **is less than**
   - For **Value**, select **3**
5. Click the green plus sign to add another rule.

6. **Rule 3**:
   - For **Column**, select **Alert Message**
   - For **Condition** select **contains**
   - For **Value**, enter `openvault`
7. Click **Filter** to apply the rules. The display would then reduce to that shown in Figure 6-19.

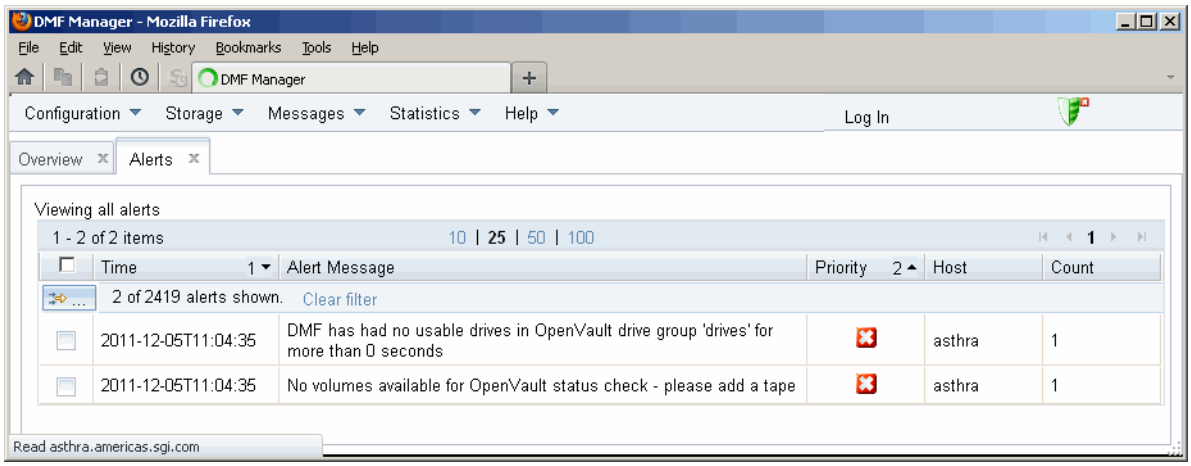

Figure 6-19 Filtered Alerts

Seeing Relationships Among DMF Components

To see the relationships among DMF components, click on a component icon in the **Overview** panel and select its **Show Relationships** menu item. Figure 6-20 shows the relationships for the `ftp1` FTP MSP.

To remove the relationship lines, click **Hide Relationships**.

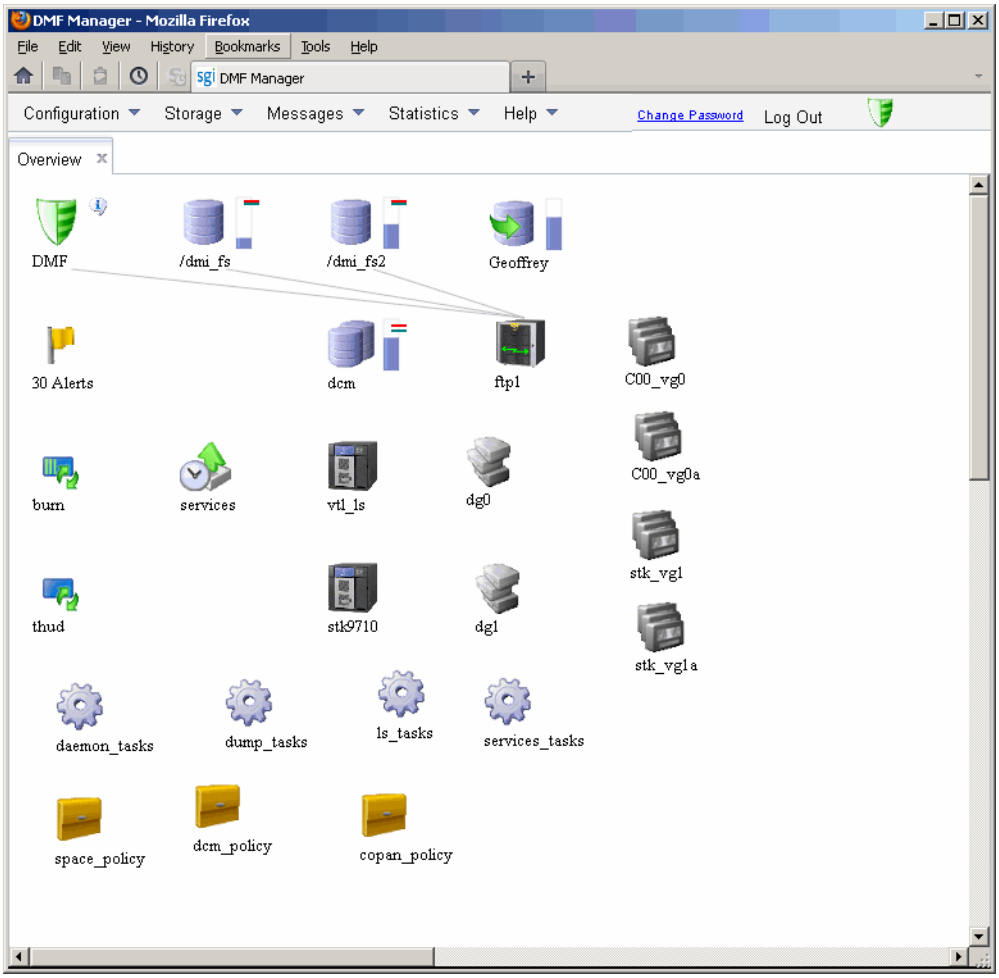

Figure 6-20 Relationships Among DMF Components

Managing Volumes

To manage volumes, select the following:

Storage
> Volumes

Figure 6-21 shows an example.

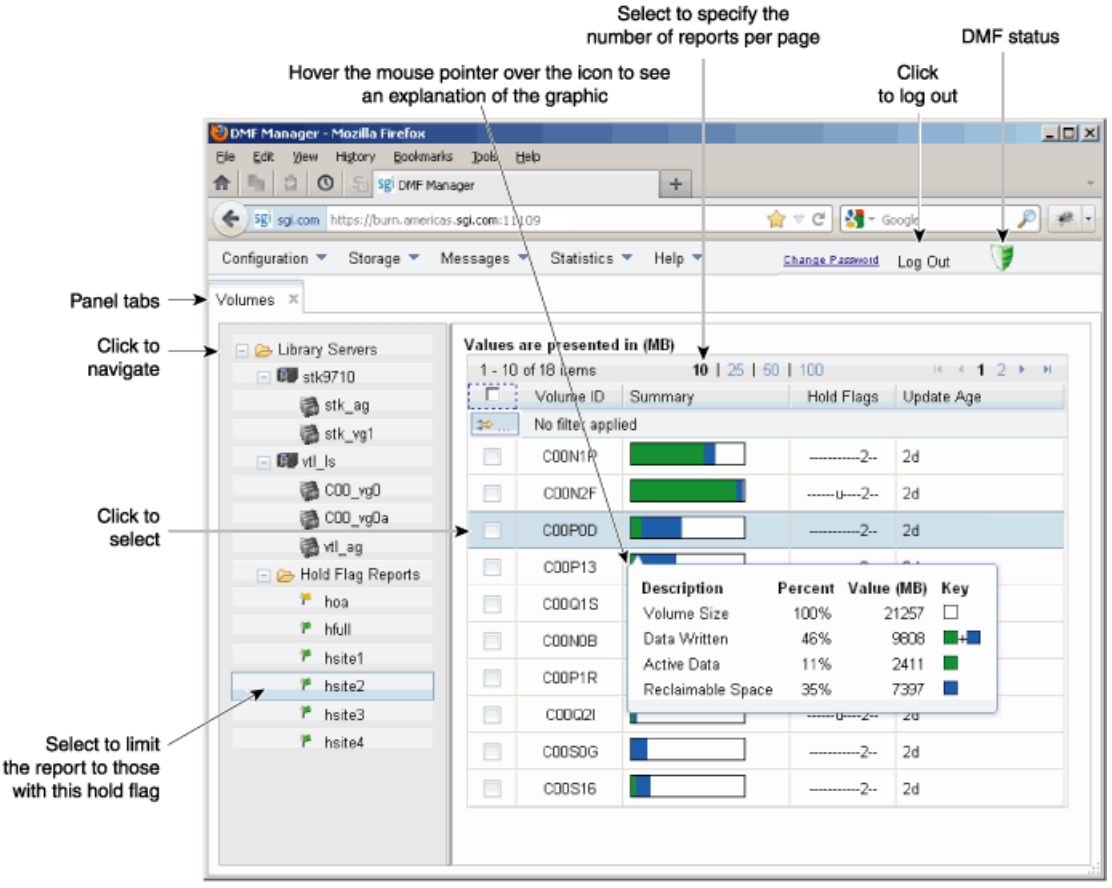

Figure 6-21 DMF Manager Volumes Panel

You can filter the volumes displayed, similar to the information in "Filtering Alerts" on page 225.

When logged in, you can also perform the following actions for selected volumes:

- **Change the Hold Flag (hflag)**, shown in Figure 6-22, sets the hold flag values on individual volumes. Click the **On** column to enable a flag or click the **Off** column to disable a flag. For more information about the hold flags, click the **Help** button or select the **What is** menu for the flags displayed in the **Volumes** panel.

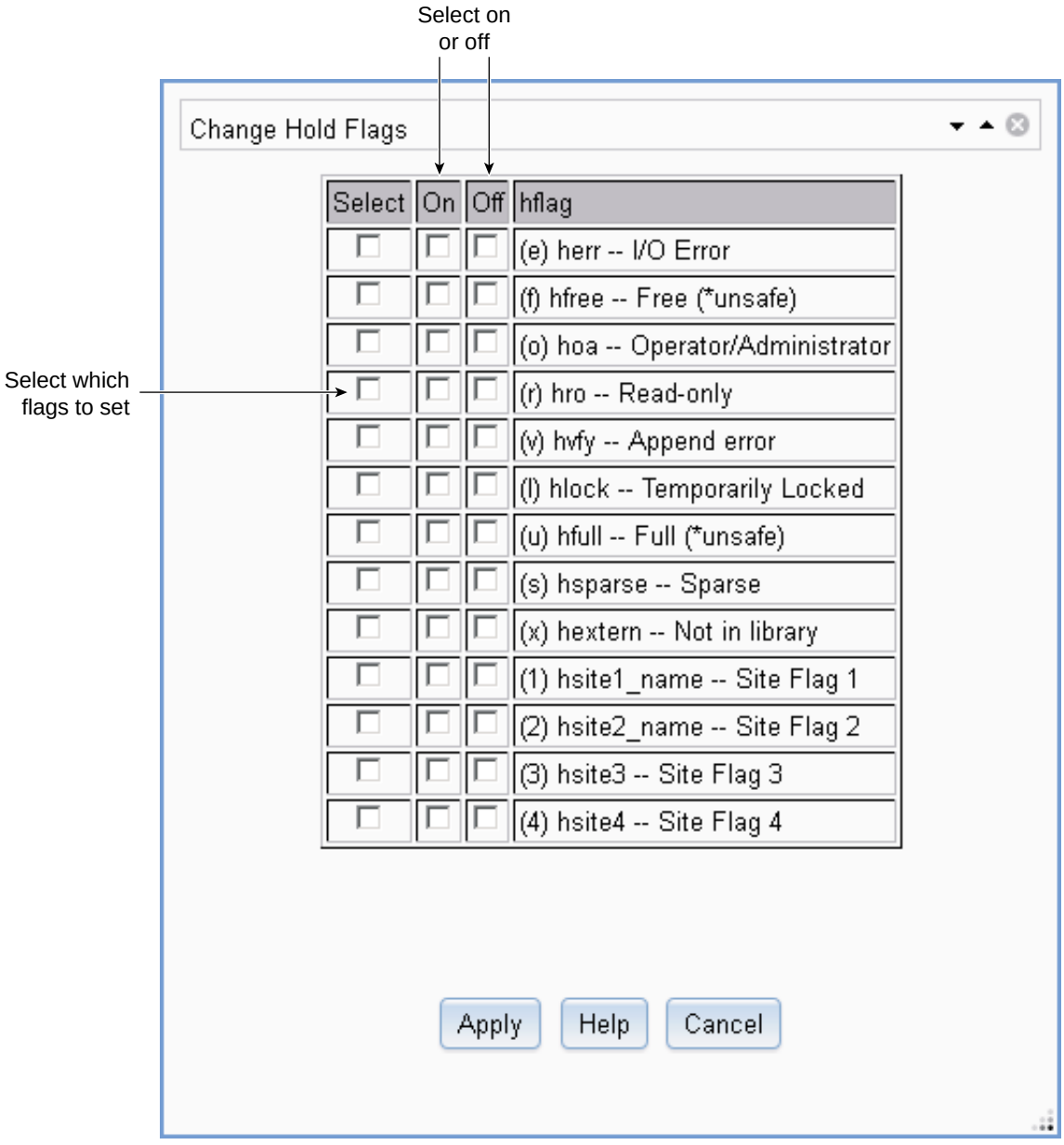

Figure 6-22 Changing Hold Flags in DMF Manager

- **Merge Data Off Volume** marks a volume as a candidate to be merged with another volume, thereby recovering space that was lost due to holes in the volume from deleted data (a *sparse volume*). These operations will be performed when appropriate. This is the preferred way to move data off of a volume.

  Note: Merging is not appropriate for a volume configured as a fast-mount cache.

- **Empty Damaged Volume** forces data to immediately move to another volume.

  Note: Use this as a last resort. You should first try **Merge Data Off Volume**.

- **Eject Tape** removes the selected physical tape cartridges from the tape library but keeps their tape IDs (*volume serial numbers*, or *VSNs*) in the VG. (In some cases, this command may cause a door to be unlocked, requiring a human operator to physically extract the cartridge from the library.) This only applies to physical tapes managed by OpenVault.
- **Verify Volume Integrity** runs a verification to make sure that the data on the volume is readable.

See "Running Observer Mode or `admin` Mode" on page 195.

You can also use the following menu bar selection to add volumes that are managed by OpenVault:

Volumes
> Add Volumes ...

Managing Libraries

To view the status of libraries, choose the following from the menu bar:

Storage
> Libraries

If you are logged in to DMF Manager, you can enable or disable the selected libraries. See "Running Observer Mode or `admin` Mode" on page 195.

Displaying DMF Manager Tasks

A given DMF Manager task may require issuing a set of DMF commands, and these commands may take some time to execute. The following panel displays the long-running DMF Manager tasks that have been issued but not yet acknowledged:

Messages
> DMF Manager Tasks

When logged in, you can choose to show the tasks logs or acknowledge, suspend/resume, or kill each selected DMF command, as appropriate. See "Running Observer Mode or `admin` Mode" on page 195.

Monitoring DMF Performance Statistics

The **Statistics** menu provides current and historical views of DMF activity and resources. This section discusses the following:

- "Using the Statistics Panels" on page 234
- "Metrics Collection" on page 235
- "DMF Activity" on page 235
- "DMF Resources" on page 238
- "DMF I/O" on page 248

Note: To see all of the available statistics via DMF Manager, you must set the `EXPORT_METRICS` configuration parameter to `ON`. Do not change this parameter while DMF is running; to change the value, you must stop and restart DMF. See "`base` Object" on page 277.

Using the Statistics Panels

The **DMF Resources** and **DMF Activity** panels of the **Statistics** menu are divided into the following areas:

- Report tree
- Graphs
- Key

To resize an area, drag the divider lines to the left or right.

Expandable folders in the tree (such as **Requests**) contain reports (such as **Requests Summary**) and subfolders (such as **Filesystem Requests**). Click on the **+** symbol to expand a folder or on the **—** symbol to contract it, or use the **Expand All** and **Collapse All** buttons. Click on a report name to display the associated graphs.

Each graph is scaled according to the maximum value in each graph. To scale all of the graphs with a common maximum value, check **Scale graphs equally** at the top of the tree.

White space within a graph means that nothing happened during that time period, or data was unavailable. This does not indicate an error condition.

DMF Manager distinguishes between the following:

- *Current metrics* are either drawn live from the server or are taken from the last few minutes of the metric archives
- *Historic metrics* are taken exclusively from the metric archives

DMF Manager is able to display historical information for the following time periods:

- Last hour
- Last day (the previous 24 hours)
- Last month (the previous 30 days)

Note: Some DMF configuration parameters use multipliers that are powers of 1000, such as KB, MB, and GB. However, the **DMF Activity**, **DMF Resources**, and **DMF I/O** panels use multipliers that are powers of 1024, such as KiB, MiB, and GiB. In particular, this means that 1 MiB/s is 2^{20} = 1048576 bytes per second.

Metrics Collection

SGI Performance Co-Pilot™ continuously gathers performance metrics for the **DMF Activity** and **DMF Resources** panels. See "Monitor the Size of the PCP Metrics Archive" on page 130.

The DMF data movers (the DMF server and any parallel data-mover nodes) collect the metrics displayed in the **DMF I/O** panel. See "Monitor the Size of the PCP Metrics Archive" on page 130.

DMF Activity

This section discusses the following:

- "Overview of DMF Activity Reports" on page 235
- "Key to DMF Activity Reports" on page 236
- "Example of DMF Activity Report" on page 237

Overview of DMF Activity Reports

The reports in the **DMF Activity** panel show user-generated DMF activity:

- **Requests** reports show the number of requests being worked on
- **Throughput** reports show the rate of data throughput resulting from those requests

---

Note: Values shown are averaged over the previous few minutes, so they are not necessarily integers (as would be expected). This process also causes a slight delay in the display, which means that the values of **DMF Activity** reports do not necessarily match the current activity on the system as seen in the DMF log files.

The following types of requests are reflected in these reports:

- Requests from the user to the DMF daemon. These are presented as an aggregate across the DMF server, and on a per-filesystem basis, using the label of **Filesystem**.
- Requests from the DMF daemon to the subordinate daemons that manage the secondary storage (a *back-end request*).

Sometimes, there is a 1:1 correspondence between a daemon request and a back-end request (such as when a file is being recalled from secondary storage back to the

managed filesystem), but this is frequently not the case. For example, migrating a newly created file to secondary storage will result in one back-end request per copy, but deleting a migrated file results in a single daemon request but no back-end request at that time. Volume merges may cause a lot of activity within a VG but none at the daemon level.

In the **Summary** reports, the different types of requests are not distinguished from each other. However, if you zoom in (via one of the subfolders, such as **DCM MSP**), the resulting report shows the broad categories as well as by filesystem or by secondary storage group, as appropriate.

Note: Some DMF configuration parameters use multipliers that are powers of 1000, such as KB, MB, and GB. However, the **DMF Activity** and **DMF Resources** panels use multipliers that are powers of 1024, such as KiB, MiB, and GiB. In particular, this means that 1 MiB/s is 2^{20} = 1048576 bytes per second.

Key to DMF Activity Reports

Each report under the **DMF Activity** tab shows an instantaneous pending-requests graph and history graphs showing the following color-coded amounts of pending requests:

Note: The exact definitions vary by report. For more a more precise description, click on an individual label.

- Summary reports:

Key	Description
Filesystem requests	Number of all daemon requests that are pending
VG & MSP requests	Number of VG and MSP requests that are pending
Last hour average	Marker that shows the average number of pending requests during the last hour
Last day average	Marker that shows the average number of pending requests during the last 24 hours

- Aggregate and individual reports:

Key	Description
Administrative	Number of daemon requests that are pending or throughput for such requests
Migrations	Number of migration-related requests that are pending or throughput for such requests
Recalls & copies	Number of requests to recall/copy data or throughput for such requests
Merges	Number of merge requests that are pending or throughput for such requests (for VGs only)
Other user activity	Number of other requests related to user actions (such as daemon remove requests or DCM cancel requests) or throughput for such requests

Example of DMF Activity Report

Figure 6-23 is an example of a filesystem throughput report. It shows that the primary activity for the `/dmfusr` filesystem are migrations, with a smaller number of recalls and copies.

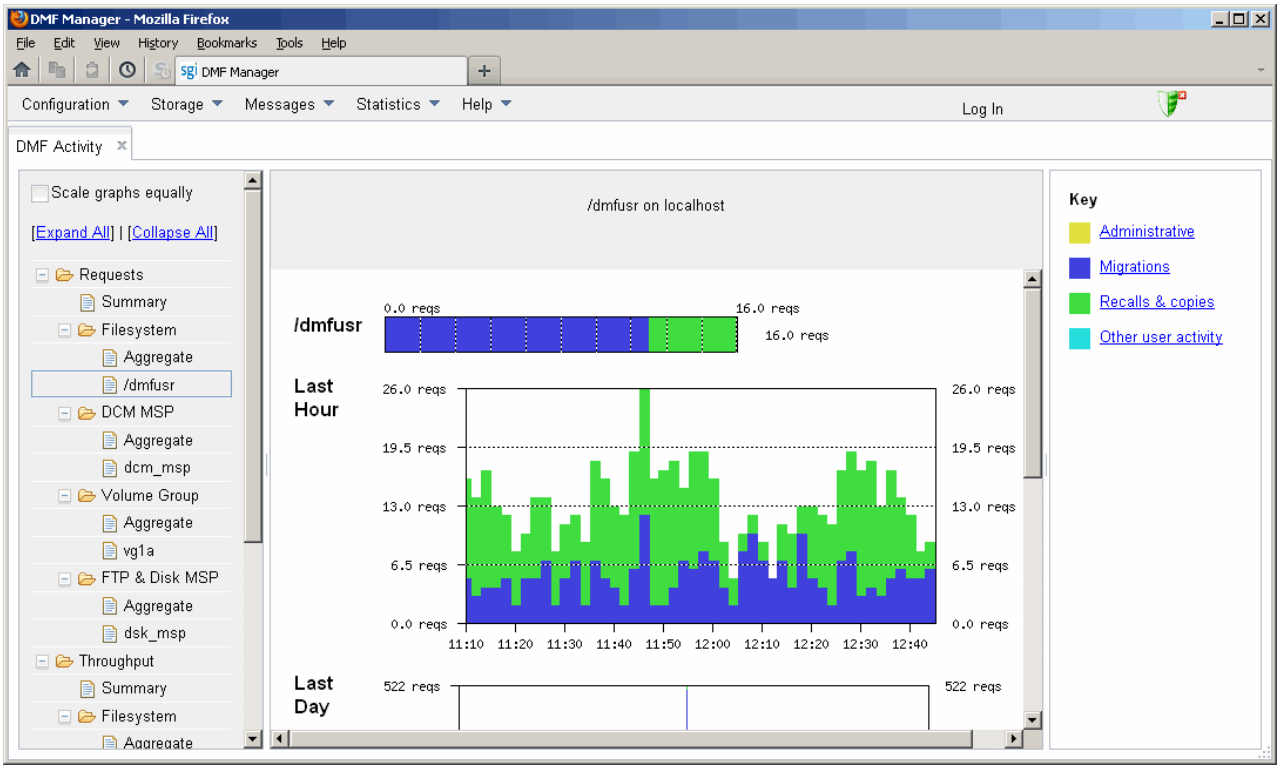

Figure 6-23 DMF Activity

DMF Resources

The **DMF Resources** panel shows how DMF is using its filesystems and hardware, as described in the following sections:

- "Programs that Update the DMF Resources Reports" on page 239
- "Filesystem Folder" on page 239
- "Libraries Report" on page 242
- "Drive Group Folder" on page 242
- "Volume Group Folder" on page 244

- "DCM MSP Folder" on page 246

Programs that Update the DMF Resources Reports

The reports in the **DMF Resources** panel are updated at the interval specified in the **DMF Resources Preferences** menu item by those DMF programs that scan the filesystem inodes:

```
dmaudit
dmdadm
dmdskfree
dmfsfree
dmhdelete
dmscanfs
dmselect
```

Filesystem Folder

Each report under **Filesystem** shows an instantaneous occupancy graph and history graphs showing the following color-coded amounts of space in the managed filesystem:

Key	Description
Free	Free space
Not migrated	Space used by files that are not migrated, such as regular files, files that will never be migrated, and files in the process of migration
Dual- & partial-state	Space used by dual-state files (files where the data resides both on online disk and on secondary storage) and partial-state files (files where the data resides both on online disk and on secondary storage)

For more information about file states, see "File States" on page 14.

The reports also display the following values:

Offline	The amount of space used in secondary storage for files in the managed filesystem
Oversubscribed	The amount of space that is *oversubscribed*, which is a ratio of offline space to the total amount of space for a given DMF filesystem (including space that is free,

space that is occupied by regular files, space that is occupied by files that are migrated, including dual-state files), calculated as follows:

offline_space / (*free_space* + *migrated_space* + *not_migrated_space*)

Note: This is a measure of data that *could be* on disk but is not at this moment in time, rather than a measure of the total amount of secondary storage being used. The fact that a migrated file may have more than one copy on the secondary storage is not considered.

Typically, the oversubscription ratio is the range of 10:1 to 1000:1, although is can vary considerably from site to site.

The data presented in the graph is gathered periodically by DMF. The time at which this information was gathered is displayed at the top of the page. The default configuration is to update this information once daily (at 12:10 am).

Figure 6-24 is an example of a filesystem resource graph. It shows that the majority of filesystem space for the `/dmfusr` filesystem is used by dual-state or partial-state files. (White space within the graph means that data was unavailable during that time period.)

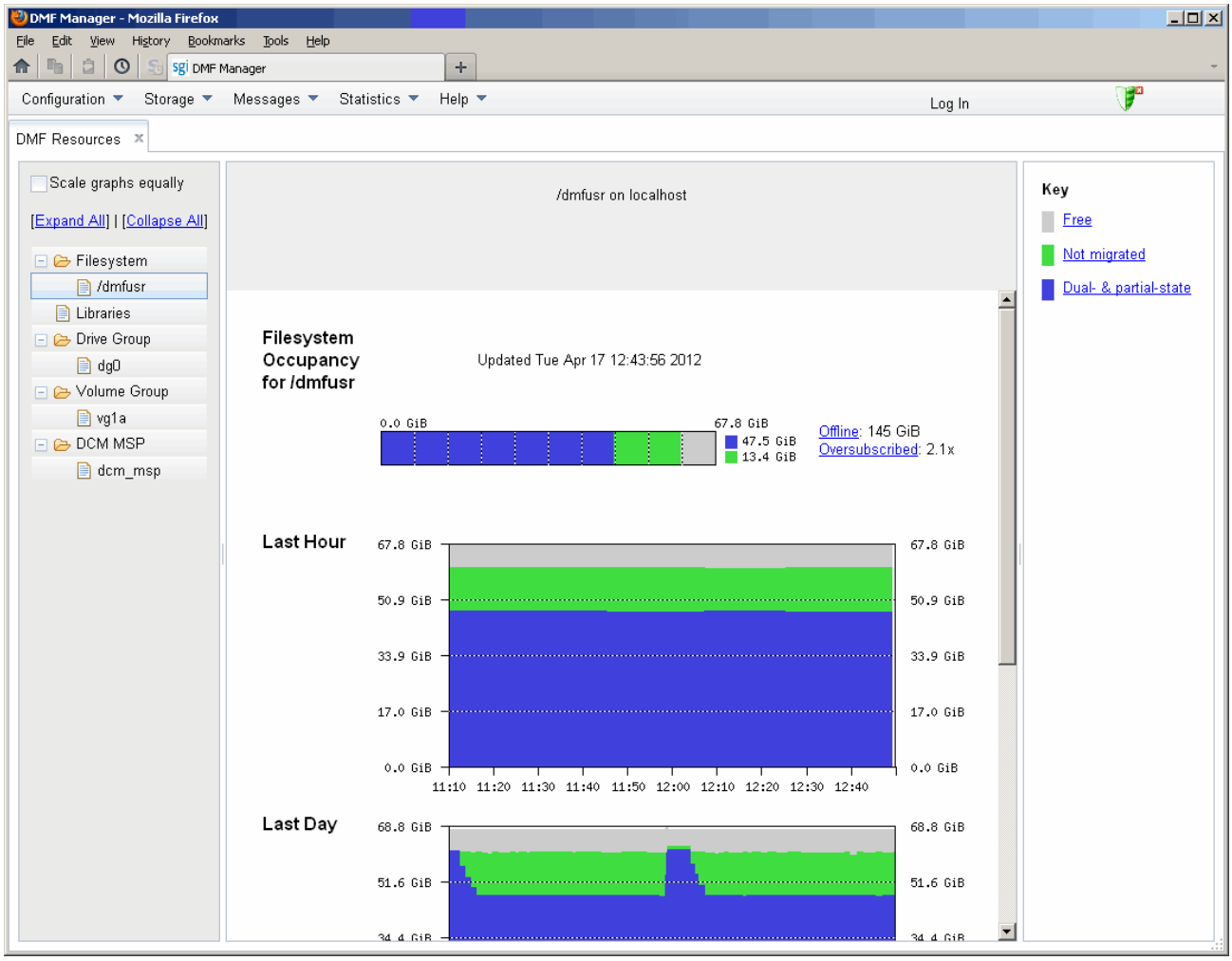

Figure 6-24 Filesystem Resource Graph

Libraries Report

Note: The **Libraries** report is available only if you are using OpenVault. This folder is unavailable if you are using TMF.

The **Libraries** report displays the number of slots that are used by DMF, used by other applications, and empty, according to information obtained from OpenVault.

Drive Group Folder

The reports in the **Drive Group** folder provide information for each drive according to the fields you select in the right-hand column:

- **Base**, which provide basic information on drive activity
- **Current**, which provide instantaneous values of drive activity and throughput
- **Total**, which provide aggregate values of drive activity and throughput
- **Averages**, which provide averaged values of drive activity and throughput

Note: This information is available only for DMF's volumes. Any other use, such as filesystem backups or direct use by users, is not shown.

To display a field in the table, click on its check box in the right-hand column. To display all fields for a given category, click on the check box for the category name, such as **Base**. For more information about a field, right-click on its column header in the table and select **What is**.

To sort according to a given column, select the up or down arrow at the upper-right corner of the column header. If you sort by multiple columns, their order is displayed in the column header. To remove sorting for a column, click on the **X** icon.

Figure 6-25 shows that drive `lto1` is in the process of mounting.

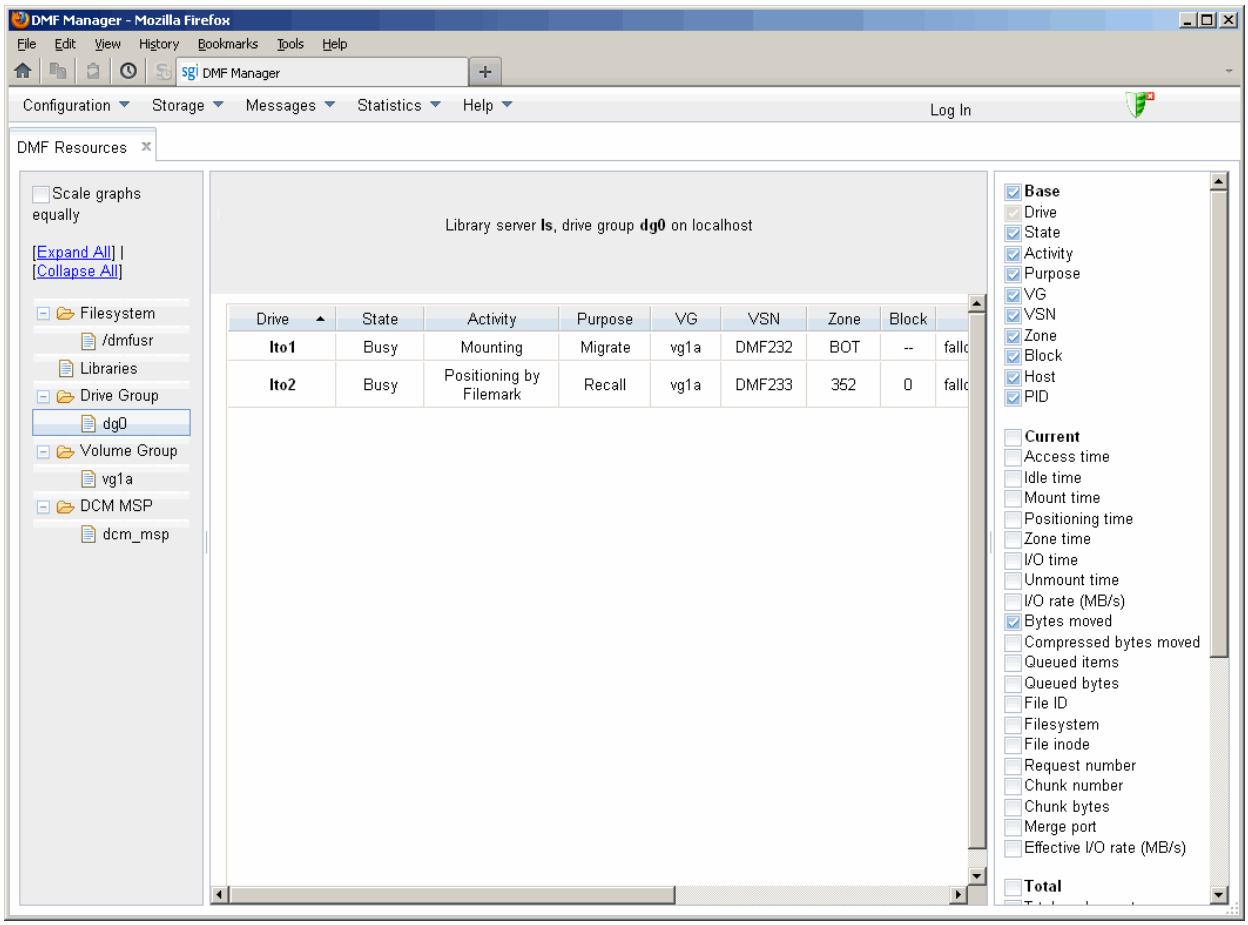

Figure 6-25 Drive Group Resource Information

Volume Group Folder

Each report under **Volume Group** shows the slot usage for this VG (for all libraries), the allocation group (AG) report (if applicable), and the volume states in an instantaneous occupancy graph and history graphs showing the following color-coded amounts of space in the managed filesystem:

The key is as follows:

Key	Metrics
Empty	Number of empty volumes assigned to DMF
Partial	Number of partially-filled volumes assigned to DMF
Merging	Number of volumes being merged
Locked	Number of volumes waiting for the `hlock` hold flag to clear
Waiting to be freed	Number of volumes waiting for the `hfree` hold flag to clear
Read-only	Number of volumes available for reads only (excluding volumes with the `hfull` hold flag set)
Unavailable	Number of volumes indefinitely unavailable (that is, those with the `hoa` operator / administrator hold flag set)

For more information about hold flags, see "`dmvoladm` Field Keywords" on page 557.

Figure 6-26 is an example of an instantaneous VG resource graph. (White space within the graph means that data was unavailable during that time period.)

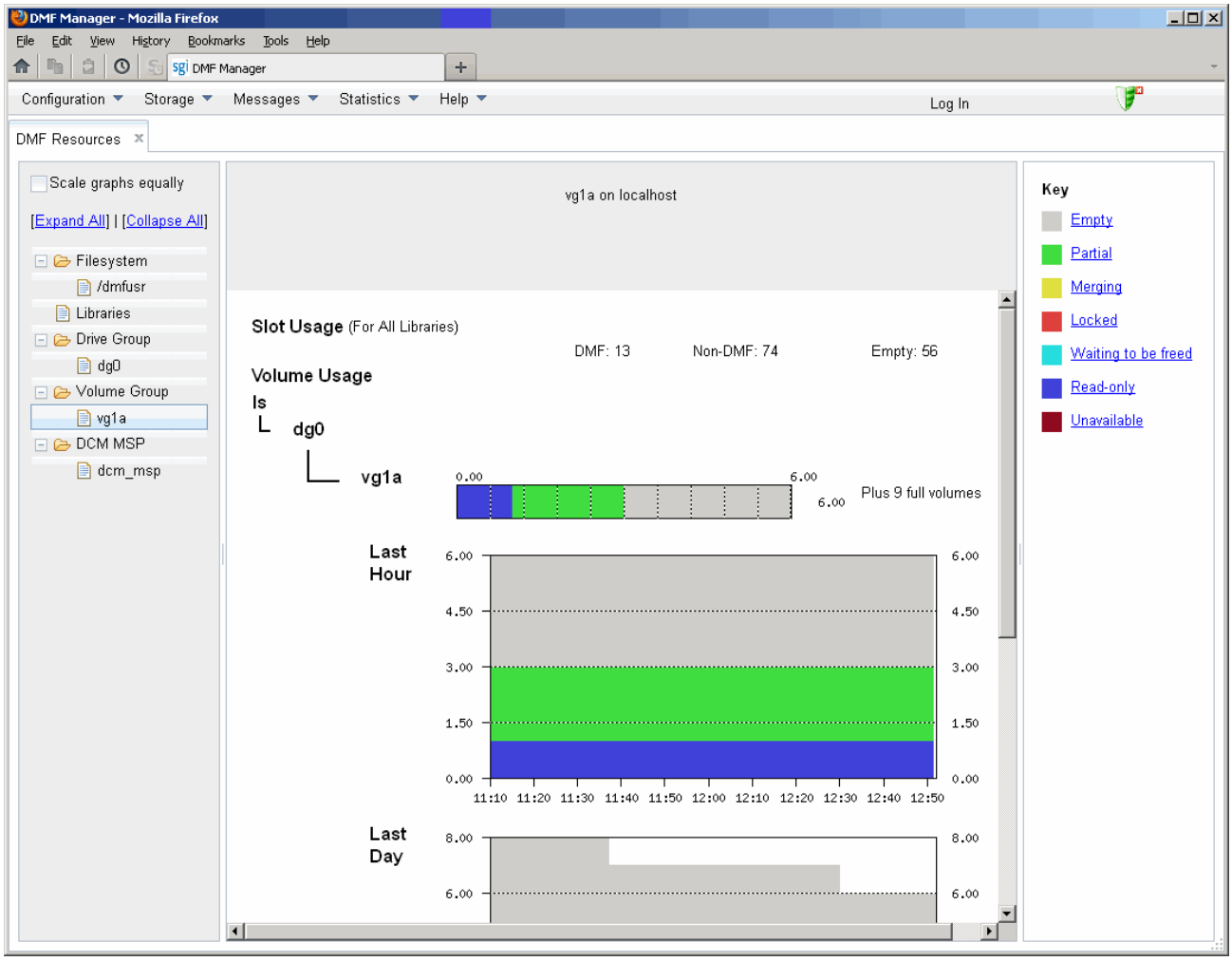

Figure 6-26 Volume Group Resource Graph

DCM MSP Folder

The reports in the **DCM MSP** folder show the DCM MSP occupancy. The key is as follows:

Key	Description
Free	Amount of space that is free in the DCM MSP *STORE_DIRECTORY* filesystem
Dual-resident	Amount of space used in the DCM MSP *STORE_DIRECTORY* by dual-resident files
Not dual-resident	Amount of space used in the DCM MSP *STORE_DIRECTORY* by files that are not dual-resident, such as incompletely moved files and files that have been completely moved to the DCM MSP *STORE_DIRECTORY* but are not in a lower VG

Note: The DCM MSP reports have similar issues to filesystem reports with regard to the frequency of updates, as described in "Filesystem Folder" on page 239.

Figure 6-27 is an example of a DCM MSP resource graph. It shows the majority of the cache disk space is not dual-resident.

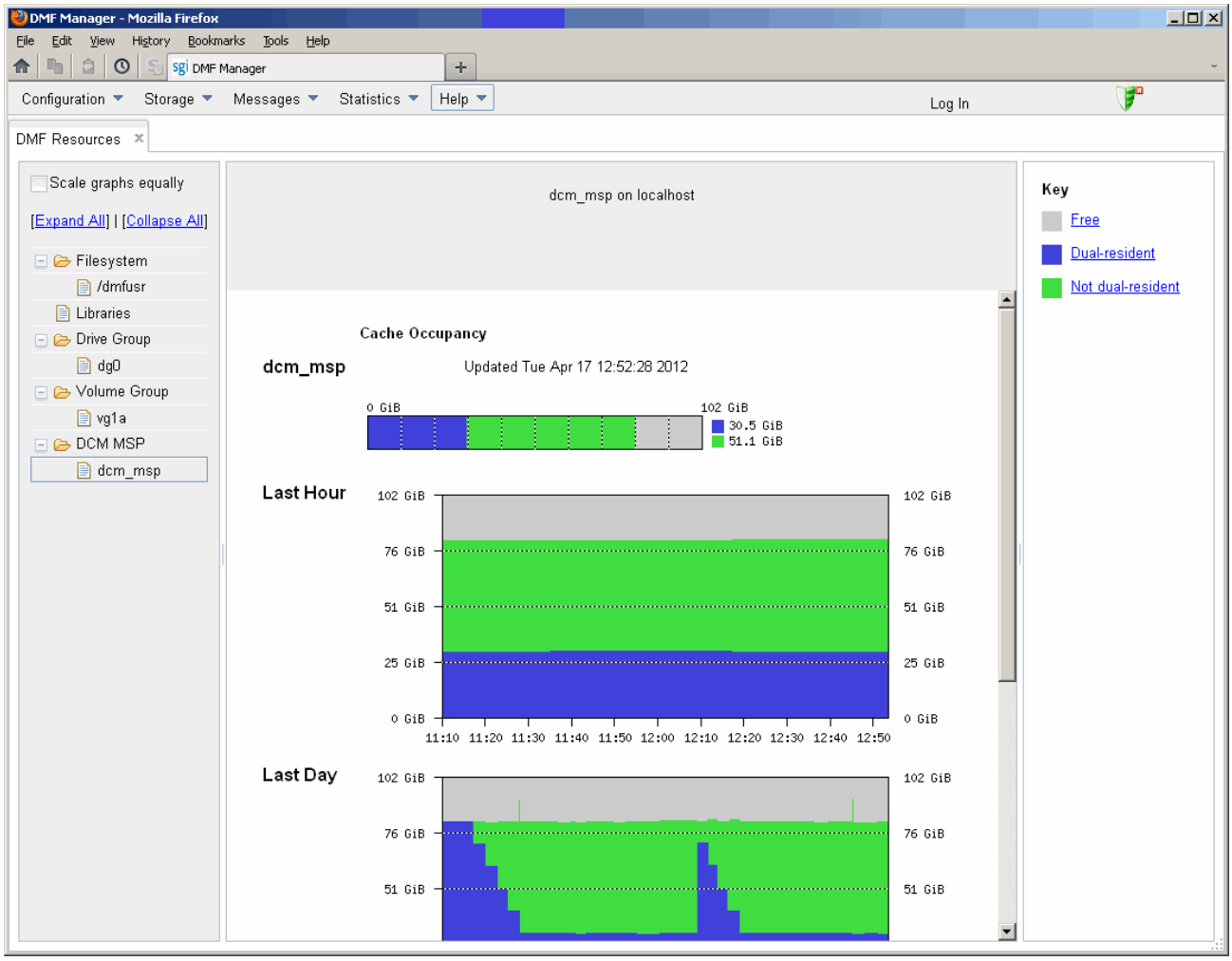

Figure 6-27 DCM MSP Resource Graph

DMF I/O

The **DMF I/O** panel lets you can create custom charts. One chart can represent multiple data items. You can group similar charts under a folder to view them on a single page. To save a chart, select the following:

DMF I/O
> Save current configuration of charts

To return to the previous configuration of charts, select the following:

DMF I/O
> Return to previous configuration of charts

Any DMF user can display, save and overwrite the chart configuration.

Note: To see I/O statistics via DMF Manager from all data movers, ensure that the `PERFTRACE_METRICS` configuration parameter is set to `ON` (the default is `OFF`). See "`base` Object" on page 277.

The **DMF I/O** panel lets you create custom charts that show how DMF is using data movers and various kinds of specific media:

- Volumes
- Drives
- Filesystems (includes archive filesystems, managed filesystems, and DMF administrative directories configured by the `CACHE_DIR`, `TMP_DIR`, and `MOVE_FS` parameters)
- Servers (potential DMF servers)
- Movers (parallel data-mover nodes)

To create a chart, do the following:

1. Click **Custom Charts** in the left-hand side of the **DMF I/O** panel. This opens the **Add Customized Chart** dialog.
2. Specify the time frame of the chart by selecting one of the following:
   - **Select time range:**

  - Enter the starting and ending dates (in *month/date/year* format, such as `2/27/2013`) or use the pull-down calendar
  - Enter the time using 24–hour format, (such as `15:59` for 3:59 PM) or use the pull-down list

- **Select last:** specify the number and unit of measure (minutes, hours, or days). For example, to specify the last three hours, enter `3` and select **hours**.

3. Specify how often the chart should be refreshed (in seconds). This value cannot be less than `60`.
4. Name the folder name that will hold the chart and the individual chart. Permitted names consist of alphanumeric characters, hyphens, underscores (no whitespace is allowed).
5. Specify the data to be included in the chart:

Note: The more information you add to a single chart, the longer it will take to draw the entire chart. To reduce clutter and increase readability and responsiveness, specify no more than five data items.

You can specify the following:

- **Target:** specify the type of data to be collected. You can select from the pull-down list or enter a specific name from the list.
- **Hosts:** specify the data-mover nodes and DMF server nodes from which to collect data. You can select from the pull-down list or enter a specific name from the list.
- **Read/Write/Aggregate:** For any particular target, select what type of I/O that you want to track, which can be all of the following:
  - Read rate
  - Write rate
  - Aggregate rate (sum of both read and write I/O)
- To add line to the graph, click **Add data item**. To remove lines from the graph, click their check boxes and select **Remove selected data items**.

Note: The averaging algorithm attempts to represent most idle time periods. The idle times are excluded from the averages as much as possible.

Figure 6-28 shows an example that creates a chart named `chart1` that will display write data collected for the last 60 minutes for filesystem `/dmfusr1` on the host named `vajra`.

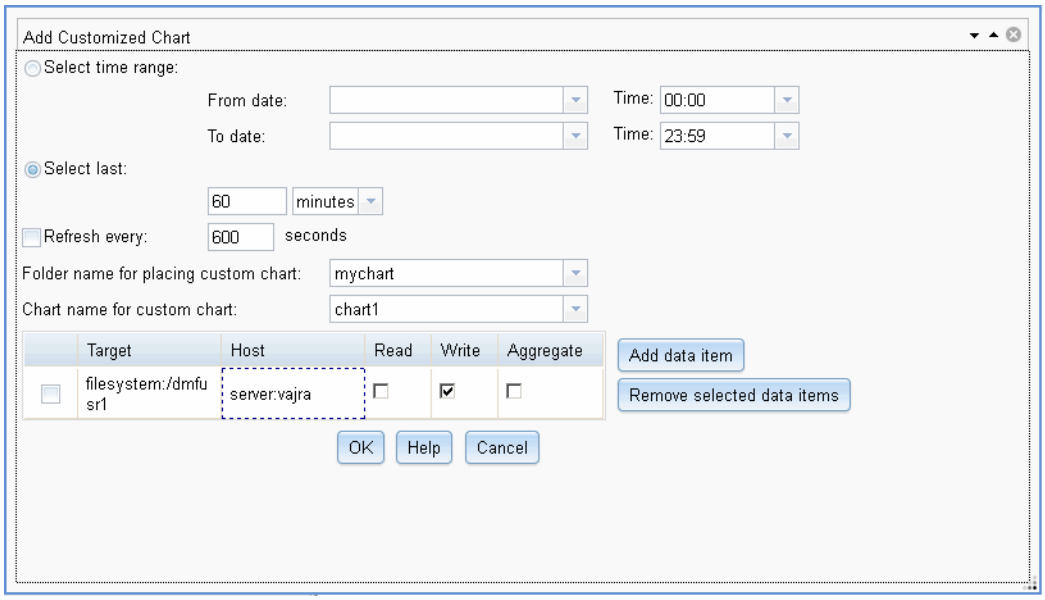

Figure 6-28 DMF I/O Custom Chart Creation

By default, the chart appears under its group name in the left-hand side of the **DMF I/O** panel, as shown in Figure 6-29.

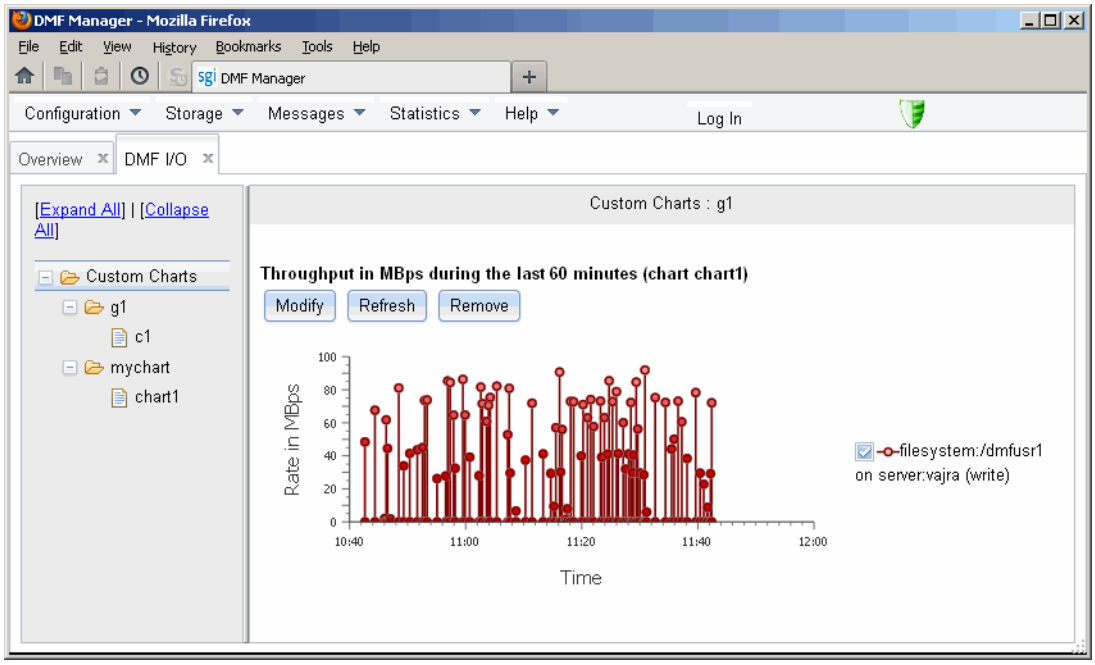

Figure 6-29 DMF I/O

To further manipulate the chart, click the **Modify**, **Refresh**, and **Remove** buttons above the chart display.

When you exit this DMF Manager session, the chart configurations you created will be removed. To save your new chart configurations, click **Save current configuration of charts** in the **DMF I/O** panel menu. To go back to the previous set of chart configurations, click **Revert to previous configuration of charts** in the panel menu (only one set is allowed).

To automatically remove old performance records, configure the `PERF_RETENTION` parameter and use the `run_remove_perf.sh` task. See "`taskgroup` Object Parameters" on page 312.

Note: An averaging algorithm attempts to represent most idle time periods. The idle times are excluded from the averages as much as possible.

Displaying Node Status

If you are running the Parallel Data-Mover Option, you can display the status of a node from the DMF and (when available) CXFS point of view by hovering the mouse pointer over the node's icon, as shown in Figure 6-30.

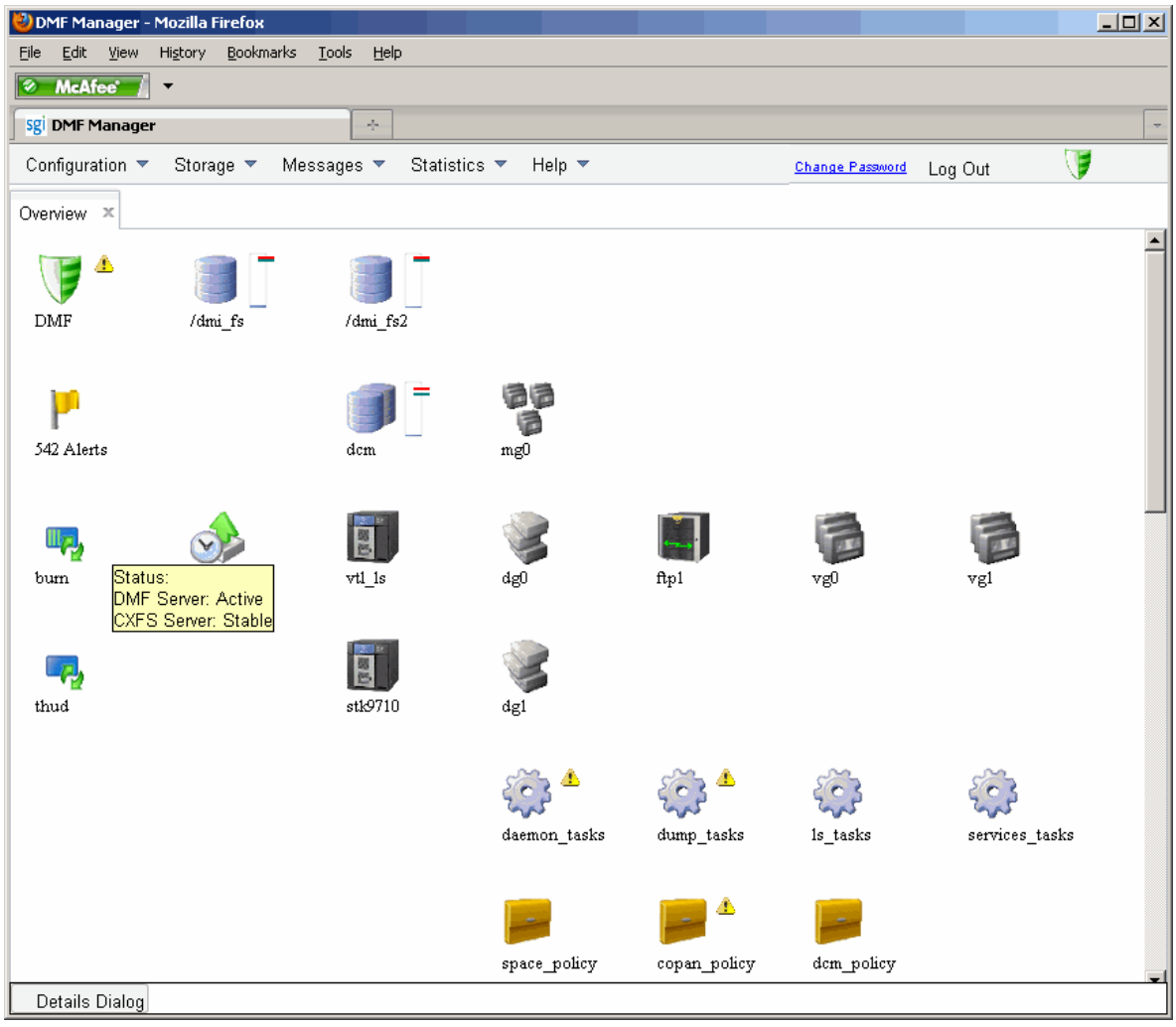

Figure 6-30 Node State

The states are as follows:

- DMF states:

  ```
  Active
  Inactive
  Disabled
  ```

- CXFS states such as:

  ```
  Stable
  Establishing membership
  Inactive
  Disabled
  ```

Right-click the icon and select **Details...** to display more information, including CXFS mount information for the DMF administrative directories and managed filesystems. Items in green font indicate that all is well; items in red font indicate a problem. Click **Help** for more information about the fields. Figure 6-31 shows an example.

Note: In a DMF HA environment, only the active DMF server is displayed.

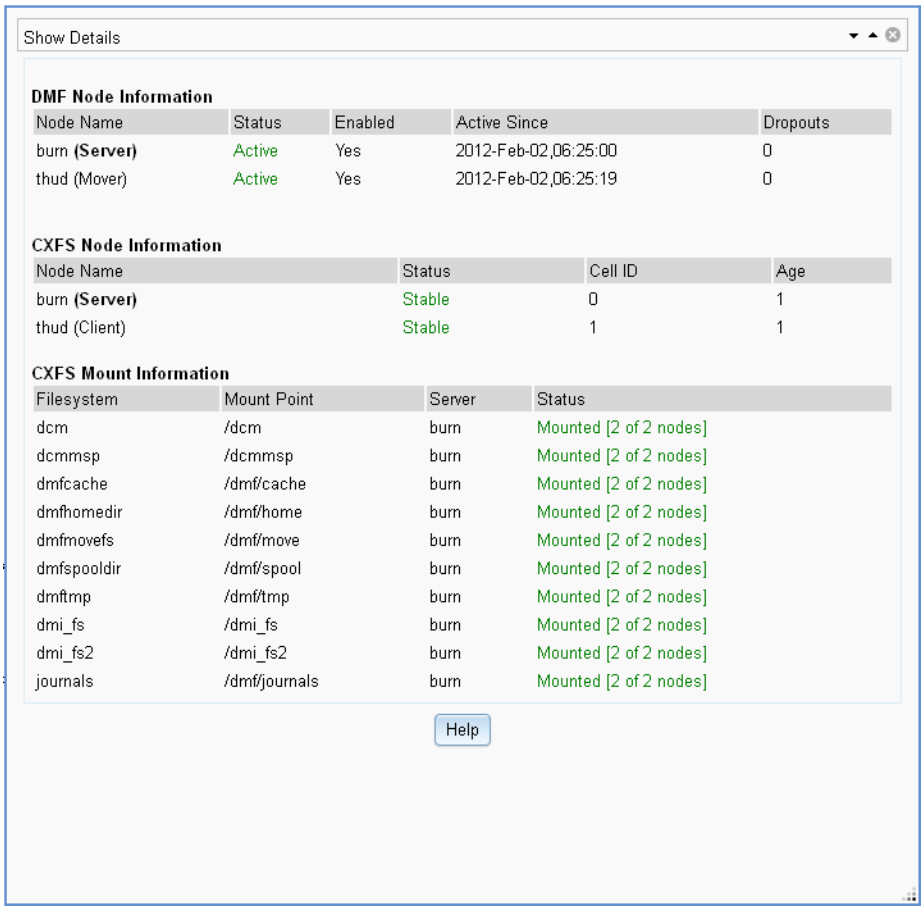

Figure 6-31 Node Details

Restoring Filesystems in DMF Manager

To restore XFS and CXFS filesystems, select the following:

Backups
> Recovery

This panel displays the available dumps (with panes for dumps produced via the integrated backup method and dumps produced via any backup method) and the status of restore processes. You must log in to complete the following administrative actions:

- "Restore a Filesystem Component (Integrated Backups)" on page 255
- "Restore a Filesystem Component (Nonintegrated Backups)" on page 260
- "Stop a Restoration Process" on page 261
- "View Restoration Logs" on page 262

Restore a Filesystem Component (Integrated Backups)

To restore a filesystem or any subcomponent from the filesystem recovery pane, do the following:

1. Click **Available Dumps (Integrated Method)** and select the desired dump.

   Note: When using integrated backups, the dumps will also be available from the **Available Dumps (All Methods)** pane, but it has fewer available actions.

2. Right-click and choose the desired action:

   - Show the dump log
   - Restore the filesystem at any level of granularity
   - Restore the DMF databases
   - Restore the `xfsdump` inventory
   - Restore configuration files

   Figure 6-32 shows an example.

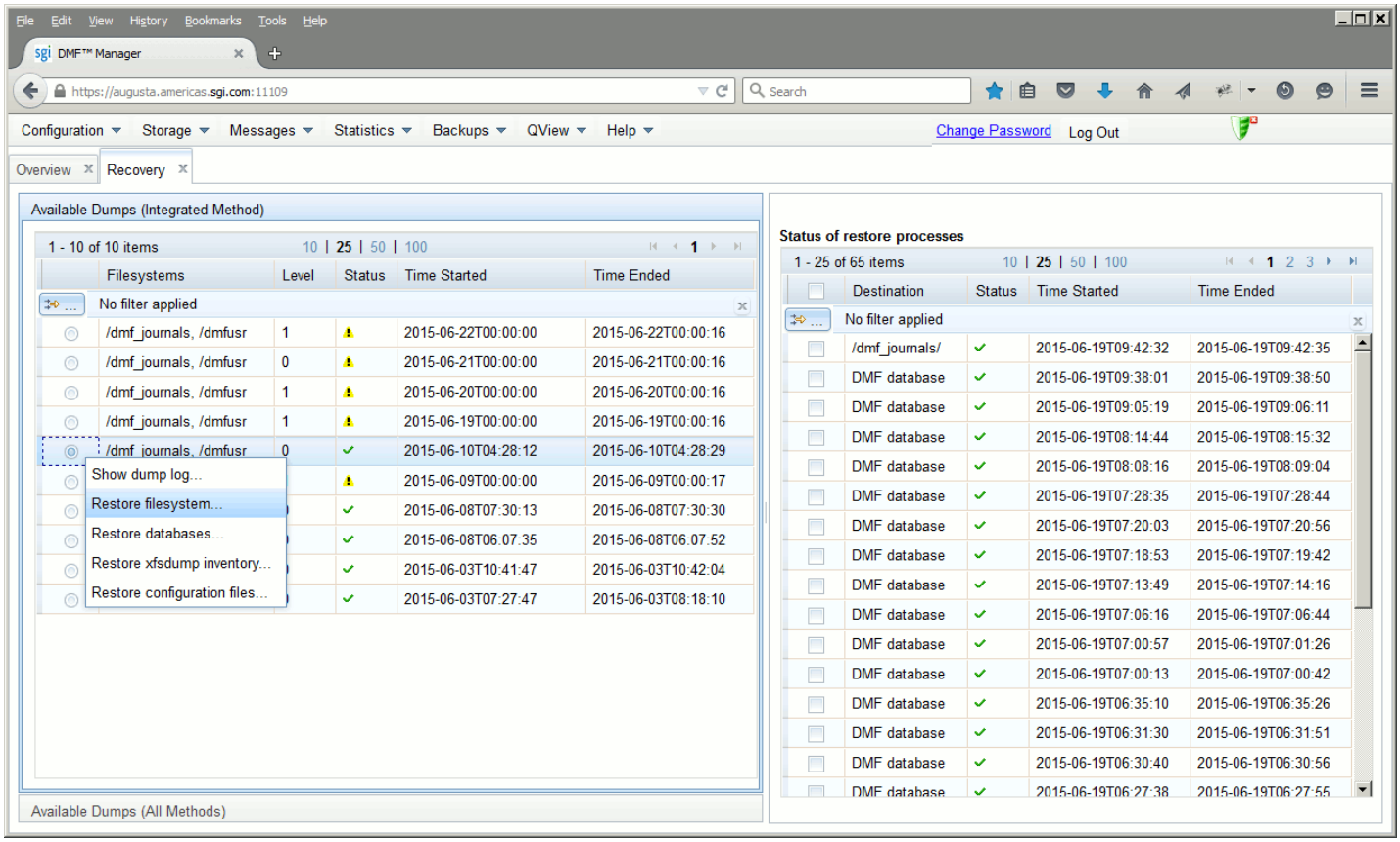

Figure 6-32 Restoring a File from **Available Dumps (Integrated Method)**

For example, the **Restore filesystem** action displays the name of the source filesystems that were included in the chosen dump, the time of the dump, the dump level, the dump method, and the status of DMF management. You can choose or supply the following:

Source filesystem

Specifies the filesystem from which to obtain data

Source directory or file relative to the filesystem

Specifies a subdirectory or the path to a file that you want to restore from the selected dump. If you want to restore the entire filesystem, leave this field empty.

Note: If the file or subdirectory is not available in this dump, an error will occur and be logged in the restore log.

Destination filesystem

Specifies the filesystem into which the restored component will be placed. By default, the name of the source filesystems is used as the destination filesystem.

Destination directory relative to the destination filesystem

Specifies the location where the component will be restored within the destination filesystem. If the path provided is not found, it is dynamically created. By default, the target will be relative to the top level of the destination filesystem.

Overwrite options

You can optionally choose to overwrite existing files or overwrite only if the source files being restored are newer than the existing files at the destination. By default, no existing files will be overwritten.

Pane behavior

You can choose to have the pane stay open until the process completes or close the pane immediately after launching the process.

The following sections show examples:

- "Example: Restoring to the Original Location" on page 258
- "Example: Restoring to a New Location" on page 260

Example: Restoring to the Original Location

For example, suppose you want to restore the `Day4` directory to its original location within the `/dmfusr1` filesystem. You would do the following:

1. Select the appropriate dump.
2. Enter `Day4` in the source field.
3. Click **Restore**

Figure 6-33 shows an example.

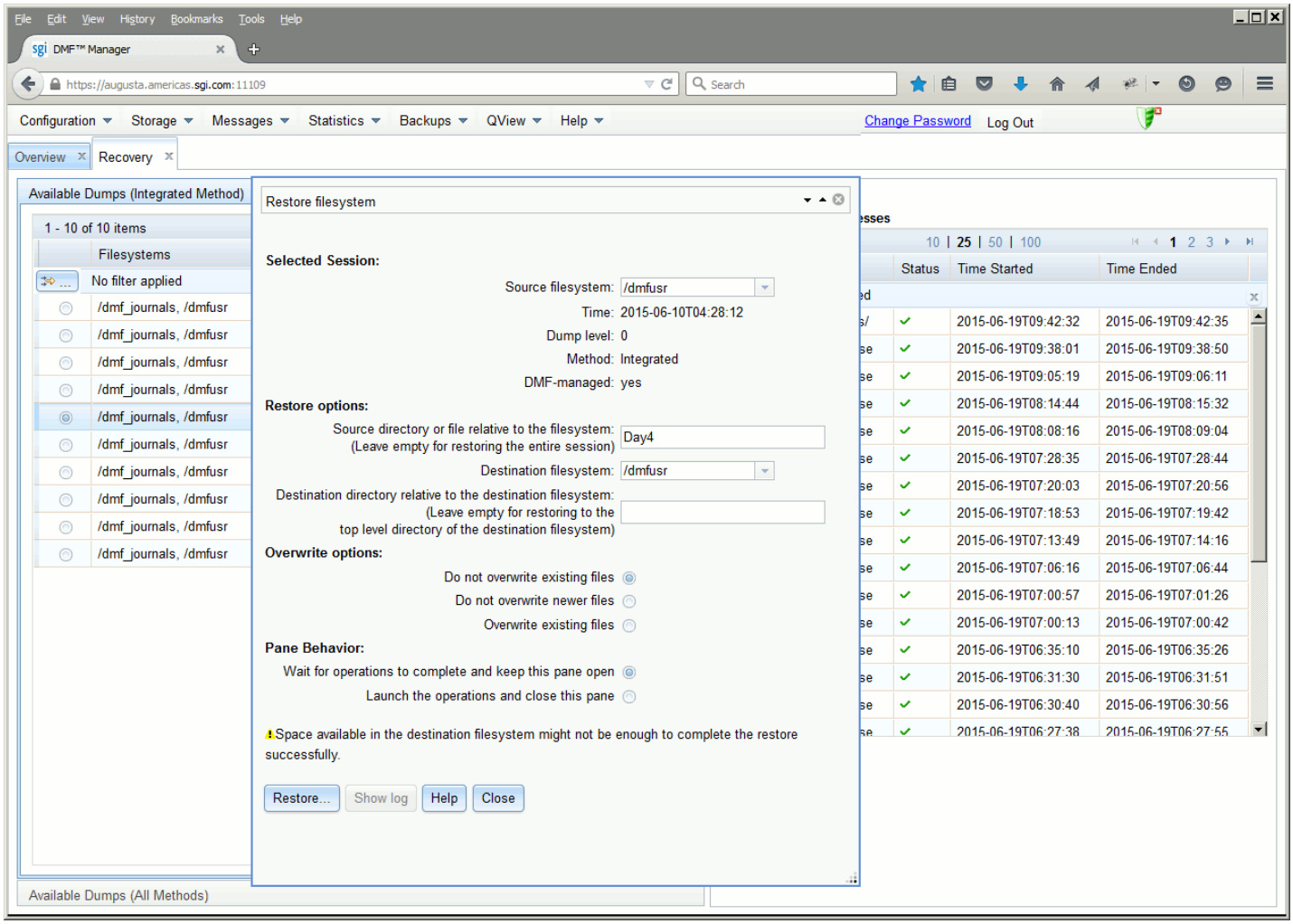

Figure 6-33 Restoring a Filesystem Component using DMF Manager (Integrated Backups)

Example: Restoring to a New Location

For example, suppose you want to restore the `Day4` directory to a new location, beneath a new directory named `/dmfusr1/newdir` that does not yet exist:

```
# pwd
/dmfusr1
# ls
Day1   Day2   Day3   Day4   Day5
```

You would do the following:

1. Select the appropriate dump.
2. Enter `Day4` in the source field.
3. Enter `newdir` in the destination directory.
4. Click **Restore**.

You would then see the following:

```
# pwd
/dmfusr1
# ls
Day1   Day2   Day3   Day4   Day5   newdir
# ls newdir
Day4
```

Restore a Filesystem Component (Nonintegrated Backups)

When using nonintegrated tape or disk backups, you can restore the filesystem at any level of granularity but you cannot perform other actions. Click **Available Dumps (All Methods)**. The filesystem component restoration procedure is similar to the above.

For example, the **Restore filesystem** action displays the name of the source filesystems that were included in the chosen dump, the time of the dump, the dump level, the dump method, and the status of DMF management. For a dump to tape, you must also supply the drive group.

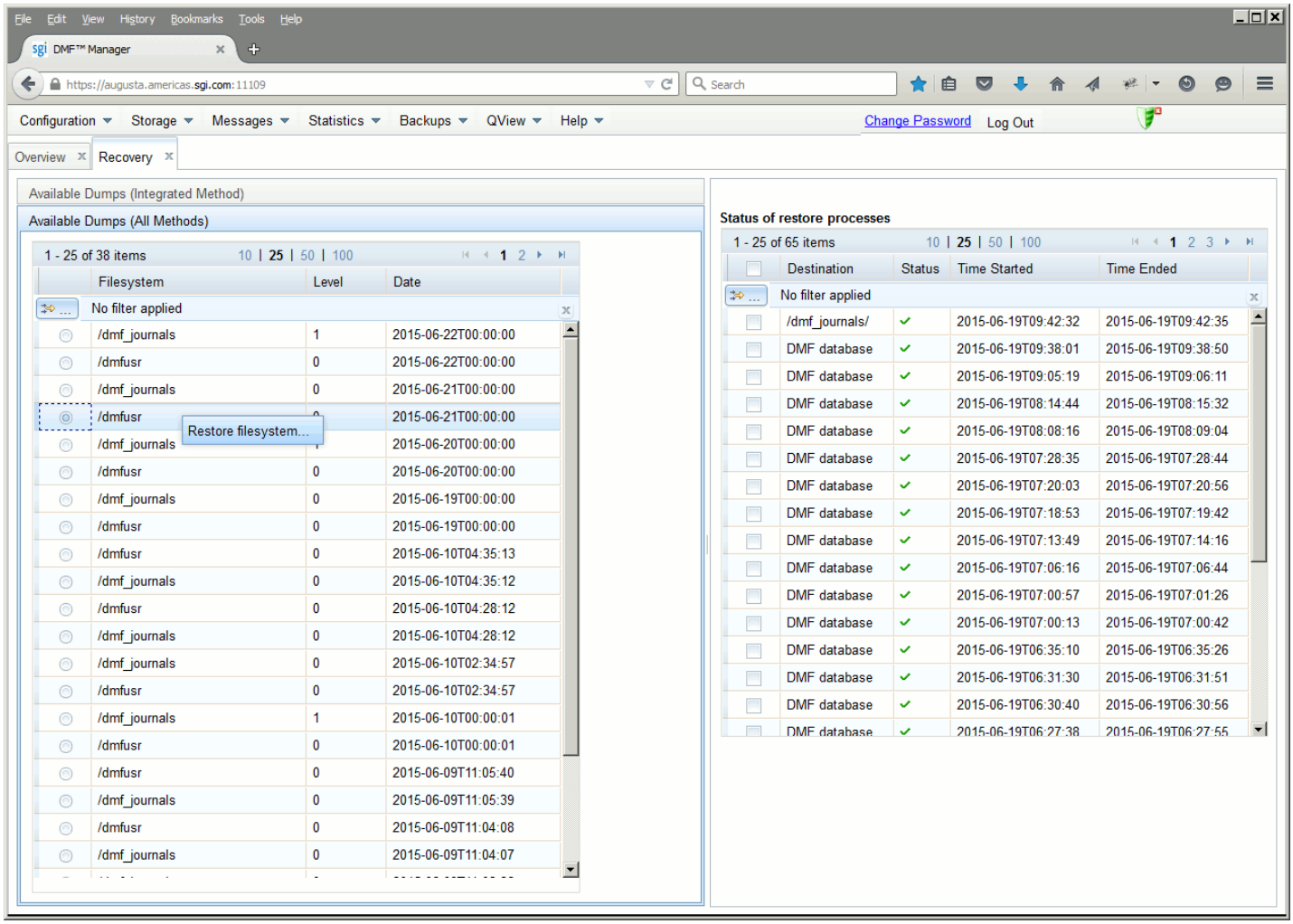

Figure 6-34 Restoring a Filesystem Component using DMF Manager (Nonintegrated Backups)

Stop a Restoration Process

To stop currently running processes, do the following in the **Status of restore processes** panel:

1. Select the check box for each the processes that you want to stop.

2. Right-click and choose **Kill**.

View Restoration Logs

To view a log for a completed process, do the following:

1. Select the check box for process.
2. Right-click and choose **Show log**.

Disaster Recovery in DMF Manager for Basic DMF

If you are using basic DMF in a non-HA environment, the following menu selection lets you perform disaster recovery tasks when you are logged in to DMF Manager

Backups
> Recovery
> Disaster Recovery ...

See "Disaster Recovery for Basic DMF without HA" on page 651.

Also see:

- "Disaster Recovery for Parallel DMF" on page 655
- "Disaster Recovery for DMF with HA" on page 657

Viewing the DMF Queue

The **QView** tab displays information about the internal DMF processing queues, equivalent to the `dmqview`(8) command. You must log in DMF Manager to use **QView**.

The **QView** page lets you directly cancel and reprioritize requests; the `dmcancel`(8) and `dmrepri`(8) commands provide the same functionality for the command line.

For details, see Chapter 17, "DMF Queue Viewing" on page 605.

LTFS in DMF Manager

This section discusses the following for the SGI Linear Tape File System (LTFS):

- "Showing LTFS Information in DMF Manager" on page 263
- "Showing LTFS Progress in DMF Manager" on page 264
- "Configuring LTFS using DMF Manager" on page 265
- "Mounting an Unassigned Tape for LTFS Use in DMF Manager" on page 267
- "Unmounting an LTFS Tape in DMF Manager" on page 269

For more information and additional functions, see Chapter 18, "SGI LTFS" on page 631 and the `sgi-ltfs`(8) man page.

Showing LTFS Information in DMF Manager

You can display information about LTFS with the following selection:

Storage
> Volumes
> Show LTFS Information

You can also view this information by clicking on **LTFS Assigned Tapes** or **Unassigned Tapes** in the navigation tree. Figure 6-35 shows an example of the information page.

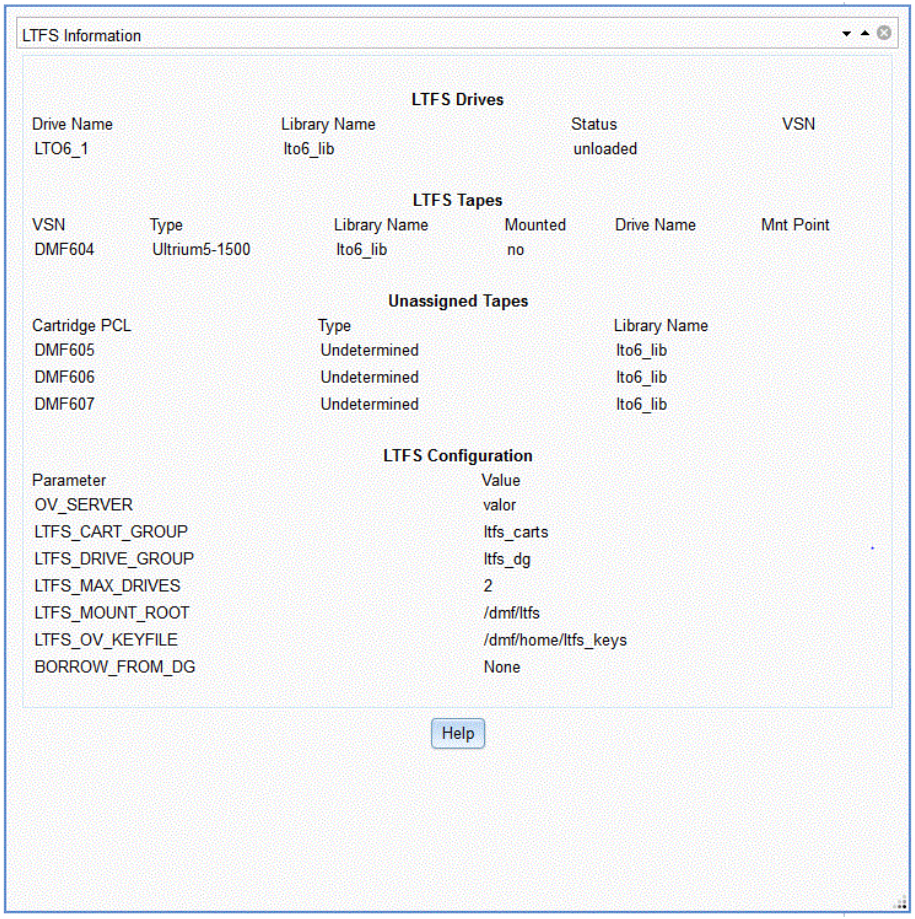

Figure 6-35 LTFS Information

Showing LTFS Progress in DMF Manager

When LTFS commands are launched from DMF Manager, they are launched as asynchronous processes. You can display their progress in the DMF Manager Tasks:

Messages
> DMF Manager Tasks

Figure 6-36 shows an example.

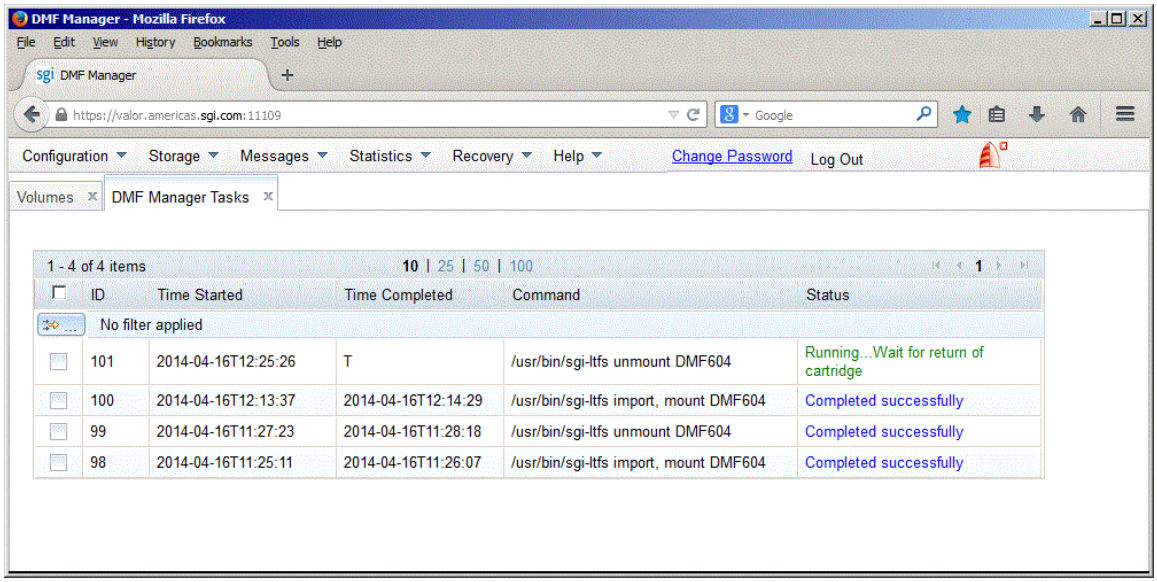

Figure 6-36 LTFS Messages

All progress and errors will be detailed in the log associated with the running command. To access the log, select a message, click, and select **Show logs**. For more information, see "Displaying DMF Manager Tasks" on page 233.

Configuring LTFS using DMF Manager

To configure LTFS, log in and use the following menu selection:

Storage
> Volumes
> Configure LTFS ...

Then enter the values you desire for the parameters shown. To get more information about a parameter, right-click on it and select the **What is this?** option.

Note: Mounting requires that LTFS has access to a drive in a valid OpenVault drive group. Valid drive groups can be reserved exclusively for LTFS use (specified by the `LTFS_DRIVE_GROUP` parameter) or can belong to another application, such as DMF (specified the `BORROW_FROM_DG` parameter), that will permit the drives to be borrowed temporarily. See "`sgi-ltfs.conf` Parameters" on page 634.

Figure 6-37 shows an example of configuring three DMF drive groups from which LTFS may borrow drives; LTFS may borrow a maximum of 1 drive from `dg1` or `dg1` (the default for a specified drive) and a maximum of 2 drives from `dg3`, with a total maximum number of 3 LTFS drives from any source.

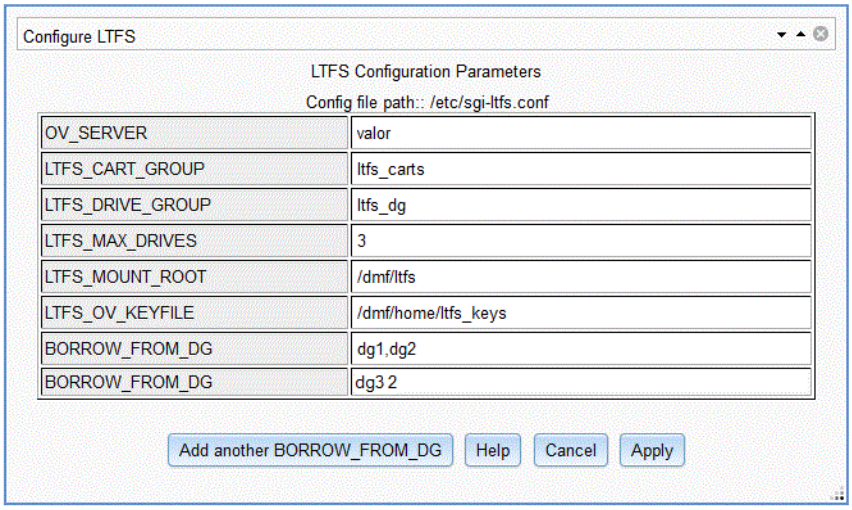

Figure 6-37 Configuring LTFS in DMF Manager

Mounting an Unassigned Tape for LTFS Use in DMF Manager

Note: Mounting requires that LTFS has access to a drive in a valid OpenVault drive group. Valid drive groups can be reserved exclusively for LTFS use (specified by the `LTFS_DRIVE_GROUP` parameter) or can belong to another application, such as DMF (specified the `BORROW_FROM_DG` parameter), that will permit the drives to be borrowed temporarily. See "Configuring LTFS using DMF Manager" on page 265 and "`sgi-ltfs.conf` Parameters" on page 634.

To mount an unassigned tape for LTFS use, log in and do the following:

1. View the unassigned tapes by using the following menu selection:

   Storage
   > Volumes

2. Click on the unassigned tape name in the left-hand tree.
3. Select the drive. By default, a valid drive will be selected for you if LTFS has been configured correctly, as noted above.
4. Select the cartridge type.
5. Specify the mount point. By default, the mount point is the same as the volume serial number (VSN).
6. Select the notification method. By default, no notification will be sent.
7. You can optionally choose to format the tape:

Warning: Formatting the tape destroys any data on the tape.

   - If the tape has already been formatted for LTFS, you can simply mount it (which by default is selected)
   - If the tape has never been formatted, you must select **Format** to mount the tape and format it for LTFS
   - If the tape has already been formatted for LTFS but you want to delete all of its data and reuse the tape, select **Force** to mount the tape and format it for LTFS
   - If you want to simply assign the tape to LTFS but not mount it or format it at this time, select **Import Only**

8. Click **Apply**. The tape will be imported and mounted or formatted, as specified above. The view will switch to the information page. To verify the completion, see "Showing LTFS Progress in DMF Manager" on page 264.

For example, Figure 6-38 shows an example of mounting an Ultrium5–1500 tape in any available drive and having email notification sent to the assigned address.

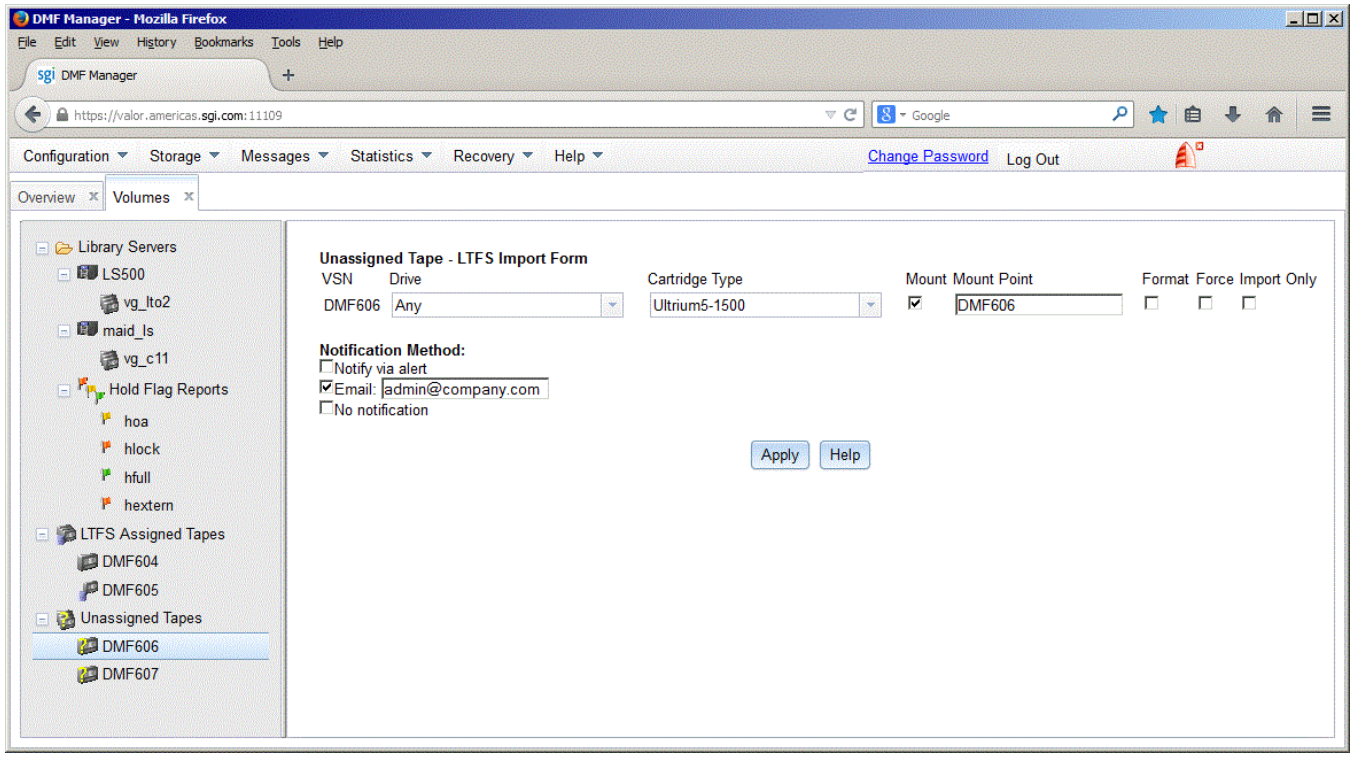

Figure 6-38 Mounting an LTFS Tape

Unmounting an LTFS Tape in DMF Manager

To unmount an LTFS tape, log in and do the following:

1. View the assigned LTFS tapes by using the following menu selection:

   Storage
   > Volumes

2. Click on the mounted tape name in the left-hand tree.

3. (Optional) To eject the tape, select the **Eject** check box. After ejection, the tape can be physically removed from the library.

4. Click **Apply and Unmount**. To verify the completion, see "Showing LTFS Progress in DMF Manager" on page 264.

Figure 6-39 shows an example.

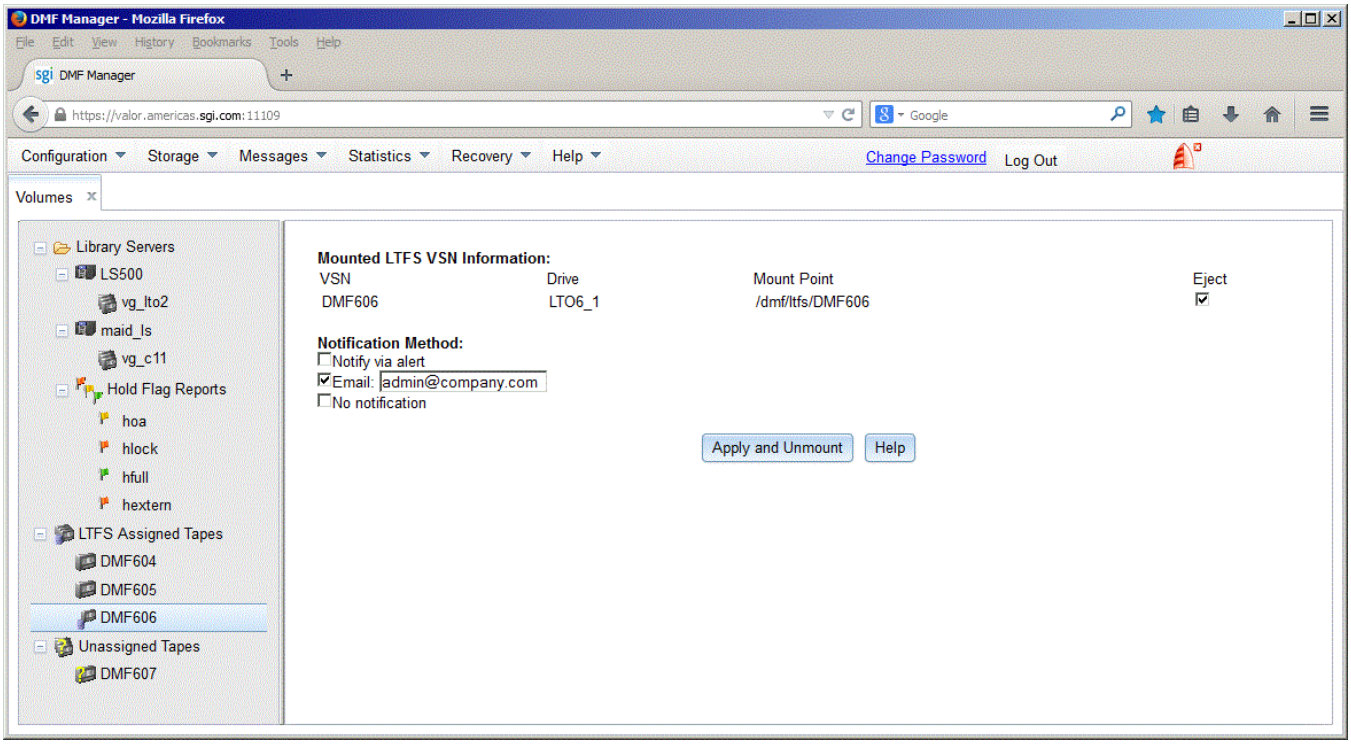

Figure 6-39 Unmounting an LTFS Tape

DMF Configuration File

This chapter discusses the following:

- "Configuration File Overview" on page 272
- "`base` Object" on page 277
- "`dmdaemon` Object" on page 289
- "`node` Object" on page 294
- "`services` Object" on page 299
- "`taskgroup` Object" on page 304
- "`dumpgroup` Object" on page 342
- "`device` Object" on page 347
- "`filesystem` Object" on page 350
- "`policy` Object" on page 364
- "`fastmountcache` Object" on page 390
- "Objects Related to an LS" on page 392
  - "`libraryserver` Object" on page 392
  - "`drivegroup` Object" on page 395
  - "`volumegroup` Object" on page 410
  - "`migrategroup` Object" on page 426
  - "`resourcescheduler` Object" on page 432
  - "`resourcewatcher` Object" on page 433
  - "`allocationgroup` Object" on page 434
  - "Examples of Configuring an LS" on page 436
  - "LS Tasks" on page 441
  - "LS Database Records" on page 444

- "msp Objects" on page 446
- "Summary of the Configuration-File Parameters" on page 472

Configuration File Overview

This section discusses the following:

- " Object Definitions" on page 272
- "Stanza Format" on page 274
- "Units of Measure" on page 275
- "Device Block-Size Defaults and Bandwidth" on page 276

Object Definitions

The DMF configuration file (`/etc/dmf/dmf.conf`) defines a set of configuration objects required by DMF. Each object is defined by a sequence of parameters and definitions; this sequence is called a *stanza*. There is one stanza for each object.

The objects defined are as follows:

- The `base` object defines pathname and file-size parameters necessary for DMF operation. See "`base` Object" on page 277.
- The `dmdaemon` object defines parameters necessary for `dmfdaemon`(8) operation. See "`dmdaemon` Object" on page 289.
- The `node` objects defines a host functioning as a data mover when using the Parallel Data-Mover Option (*Parallel DMF*). There is a `node` object for every system in the DMF configuration, excluding DMF clients. See "`node` Object" on page 294.
- The `services` object defines parameters for `dmnode_service` and other DMF services. For Parallel DMF configurations, multiple `services` objects may be defined. For basic DMF configurations, only one `services` object may be defined. (The `services` parameters all have defaults, so a `services` object is only required to change those defaults.) See "`services` Object" on page 299.
- The `taskgroup` objects define parameters necessary for automatic completion of specific maintenance tasks. See "`taskgroup` Object" on page 304.

- The `dumpgroup` object defines dump parameters that apply to a specific subset of filesystems.
- The `device` objects define parameters necessary for automatic use of tape devices. Normally, the backup scripts would refer to a DMF drive group (DG) to define parameters necessary for accessing tape drives, but if they are to use drives that are not in use by DMF, you can use a `device` object to define these parameters. See "`device` Object" on page 347.
- The `filesystem` object defines parameters related to DMF's use of that filesystem. See "`filesystem` Object" on page 350.
- The `policy` objects specify parameters to determine media-specific process (MSP) or volume group (VG) selection, automated space-management policies, and/or file weight calculations in automated space management. See "`policy` Object" on page 364.
- The `fastmountcache` object defines the migrate groups (MGs) and independent VGs (that is, those VGs that are not in an MG) that are members of the fast-mount cache. See "`fastmountcache` Object" on page 390.
- The following objects are related to a library server (LS):
  - The `libraryserver` object defines parameters relating to a library for an LS. See "`libraryserver` Object Parameters" on page 393.
  - The `drivegroup` object defines parameters relating to a pool of devices in a specific LS. See "`drivegroup` Object Parameters" on page 397.
  - The `volumegroup` object defines parameters relating to a pool of volumes mountable on the drives of a specific DG that are capable of holding, at most, one copy of user files. See "`volumegroup` Object" on page 410.
  - The `migrategroup` object defines parameters that combine a set of MSPs and VGs into an MG so that they can be used as a single destination for a migrate request. See "`migrategroup` Object" on page 426.
  - The `resourcescheduler` object defines parameters relating to the scheduling of devices in a DG when requests from VGs exceed the number of devices available. See "`resourcescheduler` Object Parameters" on page 432.
  - The `resourcewatcher` object defines parameters relating to the production of files informing the administrator about the status of the LS and its components. See "`resourcewatcher` Object Parameters" on page 434.

  - The `allocationgroup` object only applies if an `ALLOCATION_GROUP` parameter is specified in the `volumegroup` object. You should specify the `allocationgroup` stanza if you want to change the default value of its `VOL_MSG_TIME` parameter. See "`allocationgroup` Object" on page 434.

- The `msp` object defines parameters necessary for an MSP's operation. See:

  - "`msp` Object for a Cloud MSP" on page 448
  - "`msp` Object for an FTP MSP" on page 455
  - "`msp` Object for a DCM MSP" on page 459
  - "`msp` Object for a Disk MSP" on page 466

See also "Use Sample DMF Configuration Files" on page 101.

Stanza Format

A stanza has the following general format:

```
define object_name
      TYPE            object_type
      parameter       value
 ...
 enddef
```

where:

- *object_name* varies by stanza. Most names are chosen by the system administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores; they cannot begin with an underscore or contain any white space. Some require a specific name (such as `base`) and some may have longer names. All object names must be unique. See the individual object subsections in this chapter for details.

- *object_type* identifies the type. Each type has a unique type identifier, detailed in the following subsections.

- *parameter* is an argument to the object. Each object has a list of potential parameters, defined later in this chapter.

- *values* is the value of the *parameter*. Where a *value* may be a list, separate the list items by white space or tabs unless otherwise noted. If the default value of a

given parameter is appropriate for your site, you do not need to specify the parameter in the DMF configuration file.

The configuration file is case-sensitive with the exception of the following parameter values, which can appear in uppercase, lowercase, or mixed case:

ON	TRUE
OFF	FALSE
YES	1
NO	0

For simplicity, this chapter only refers to the values ON and OFF.

Lines within the configuration file can be indented for readability and the fields can be separated by spaces and/or tabs. Blank lines and all text between a hash character (#) and the end of that line are ignored. Except for comments, any line ending in a back-slash (\) continues onto the next line.

For a summary of the parameters discussed in this chapter, see Table 7-4 on page 472. For the most current set of parameters, see the dmf.conf(5) man page.

You can add site-specific parameters to any existing stanza or you can create a new stanza. You should choose parameter and stanza names that will not cause conflict with future SGI DMF parameters and stanzas. See "Start Site-Specific Configuration Parameters and Stanzas with “LOCAL_”" on page 125.

Note: Before placing a new configuration into production, it is important to verify it by running dmcheck. The dmcheck command will point out parameters and stanzas that it does not recognize.

Units of Measure

Several parameters allow you to specify the unit of measure, which can be any of the following (all of which are powers of 1000, not 1024):

k or K for thousand (10^3)
m or M for million (10^6)
g or G for billion (10^9)
t or T for trillion (10^12)
p or P for quadrillion (10^15)

Device Block-Size Defaults and Bandwidth

DMF uses the following values as the default BLOCK_SIZE value and as the bandwidth consumed in relation to the HBA_BANDWIDTH and NODE_BANDWIDTH values.

Device	Default Block Size	Bandwidth
AMPEX DIS/DST	1199840	160000000
COPAN MAID	1048576	160000000
DLT2000	131072	1250000
DLT4000	131072	1500000
DLT7000	131072	5000000
DLT8000	131072	6000000
HP ULTRIUM 2	262144	15000000
HP ULTRIUM 3	524288	80000000
HP ULTRIUM 4	524288	120000000
HP ULTRIUM 5	524288	140000000
HP ULTRIUM 6	524288	160000000
IBM 03590B1A	16384	90000000
IBM 03590E1A	32768	13500000
IBM 03590H1A	16384	13500000
IBM 03592E05	131072	100000000
IBM 03592E06	262144	160000000
IBM 03592E07	524288	250000000
IBM ULTRIUM-TD1	131072	15000000
IBM ULT3580-TD1	131072	15000000
IBM ULTRIUM-TD2	262144	30000000
IBM ULT3580-TD2	262144	30000000
IBM ULTRIUM-TD3	262144	80000000
IBM ULT3580-TD3	262144	80000000
IBM ULTRIUM-TD4	524288	120000000
IBM ULT3580-TD4	524288	120000000
IBM ULTRIUM-TD5	524288	140000000
IBM ULTRIUM-HH5	524288	140000000
IBM ULT3580-TD5	524288	140000000
IBM ULT3580-HH5	524288	140000000
IBM ULTRIUM-TD6	524288	160000000
IBM ULTRIUM-HH6	524288	160000000

```
IBM ULT3580-TD6    524288       160000000
IBM ULT3580-HH6    524288       160000000
JBFS               1048576      160000000
QUANTUM SuperDLT1  131072       11000000
QUANTUM SDLT320    131072       16000000
QUANTUM SDLT600    131072       36000000
SEAGATE ULTRIUM    262144       16000000
SONY SDX-700C      131072       12000000
SONY SDZ-100       131072       30000000
SONY SDZ-130       262144       30000000
SONY SDZ-200       524288       45000000
SONY SDZ-230       524288       45000000
STK 9840           126976       10000000
STK T9840B         126976       19000000
STK T9840C         262144       30000000
STK T9840D         262144       30000000
STK T9940A         262144       10000000
STK T9940B         262144       30000000
STK T10000A        524288       120000000
STK T10000B        524288       120000000
STK T10000C        524288       240000000
STK T10000D        524288       300000000
Other devices      65536        160000000
```

base Object

This section discusses the following:

- "base Object Name" on page 277
- "base Object Summary" on page 278
- "base Object Parameters" on page 278
- "base Object Examples" on page 286

base Object Name

The name of the base object must be base.

base Object Summary

Following is a summary of the potential parameters for the base object:

```
base
        TYPE
        ADMDIR_IN_ROOTFS
        ADMIN_EMAIL
        DB_SERVER_BIN
        DB_SERVER_HOST
        DB_SERVER_PORT
        DIRECT_IO_MAXIMUM_SIZE
        EXPORT_METRICS
        HBA_BANDWIDTH
        HOME_DIR
        JOURNAL_DIR
        JOURNAL_SIZE
        LICENSE_FILE
        METRICS_RETENTION
        NODE_BANDWIDTH
        OV_KEY_FILE
        OV_SERVER
        PERFTRACE_METRICS
        SERVER_NAME
        SPOOL_DIR
        TMP_DIR
        VALID_ROOT_HOSTS
```

base Object Parameters

The base object's parameters define pathnames and file sizes necessary for DMF operation. It is expected that you will modify the pathnames, although those provided will work without modification. All pathnames must be unique.

Warning: Never change pathnames or server names in base object parameters while DMF is running; making changes of this type can result in data corruption or data loss.

Parameter	Description
TYPE	Specifies base (required name for this type of object). There is no default.
ADMDIR_IN_ROOTFS	Specifies which DMF administrative directories and store directories can reside in the root (/) filesystem. By default, the DMF daemon does not permit a DMF administrative directory or disk MSP store directory to reside in the root filesystem, which avoids the situation where a misconfigured or incorrectly mounted filesystem could fill the root filesystem. You can override this default action by using the ADMDIR_IN_ROOTFS parameter to specify a list of directories. The DMF daemon will abort if the directory specified by any of the following parameters resides in the root filesystem but does not appear in the ADMDIR_IN_ROOTFS list: • CACHE_DIR • HOME_DIR • JOURNAL_DIR • SPOOL_DIR • Disk MSP STORE_DIRECTORY • TMP_DIR Do not change this parameter while DMF is running.
ADMIN_EMAIL	Specifies the e-mail address to receive output from administrative tasks (see "Automated Maintenance Tasks" on page 175). The mail can include errors, warnings, and output from any configured tasks. You can specify a list of addresses. When using Parallel DMF, data movers (the DMF server node and the parallel data-mover nodes) may send email to the ADMIN_EMAIL addresses. Therefore, choose addresses that can receive email from any data mover in the configuration.

DB_SERVER_BIN	Specifies the directory where the binaries related to the Postgres database server are installed. The default is /usr/bin.
DB_SERVER_HOST	Specifies the hostname where the Postgres database server should be configured to run. The default is localhost.
DB_SERVER_PORT	Specifies the port on which the Postgres database server should be configured to run. The default is 5432.
DIRECT_IO_MAXIMUM_SIZE	Specifies the maximum size of I/O requests when using O_DIRECT I/O to read from any managed filesystem or when migrating files down the hierarchy from the *STORE_DIRECTORY* of the disk cache manager (DCM) MSP. DIRECT_IO_MAXIMUM_SIZE is ignored for a particular filesystem or DCM MSP store when DIRECT_IO_SIZE is specified in the configuration stanza for that filesystem or DCM MSP. The legal range of values is 262144–18446744073709551615. The default is 1048576. By default, the unit of measure is bytes; see "Units of Measure" on page 275.
EXPORT_METRICS	Enables DMF's use of the common arena for use by dmstat, dmarenadump, and other commands. You can set this parameter to ON or OFF. The default is OFF. If set to OFF, some statistics in DMF Manager cannot be displayed. Do not change this parameter while DMF is running.
HBA_BANDWIDTH	*(OpenVault only)* Specifies the default I/O bandwidth capacity of an HBA port that is connected to drives on the node. The value is in bytes per second. All of the HBA ports connected to drives on a node are assumed to have the same bandwidth capacity. If HBA_BANDWIDTH is not specified anywhere, the default is 1024000000000000; for more information, see "Device Block-Size Defaults and Bandwidth" on page 276. For a complete description, see "node Object" on page 294. An HBA_BANDWIDTH value specified in a node object overrides the default value specified in the base object. Also see BANDWIDTH_MULTIPLIER in "drivegroup Object Parameters" on page 397.

HOME_DIR

Specifies the base pathname for directories in which files related to the daemon database and LS database reside. This directory must not be in a managed filesystem. By default, this value must not reside in the root filesystem (see the description of ADMDIR_IN_ROOTFS above). The best practice is for *HOME_DIR* to be the mount point of a filesystem that is used only by DMF. In this way, it is much less likely that the filesystem will become full and cause DMF to abort. If you choose to use *HOME_DIR* for storing HA files or scripts that must be visible on a failover platform, you must use naming conventions that will not likely conflict with present or future DMF files and you must ensure that the files do not cause the filesystem to become full. Performance characteristics of the *HOME_DIR* filesystem will impact DMF database transaction performance and may become a limiting factor in achievable DMF database transaction rates. When using Parallel DMF, *HOME_DIR* must either be a CXFS filesystem or be in a CXFS filesystem.

For guidelines about the size of *HOME_DIR*, see "Configure Filesystems and Directories Appropriately for DMF" on page 94.

Note: *HOME_DIR* must be on a separate physical device from *JOURNAL_DIR*.

Do not change this parameter while DMF is running.

JOURNAL_DIR

Specifies the base pathname for directories in which the journal files for the daemon database and LS database will be written. This directory must not be in a managed filesystem. By default, this value must not reside in the root filesystem (see the description of ADMDIR_IN_ROOTFS above). The best practice is for *JOURNAL_DIR* to be the mount point of a filesystem that is used only by DMF. In this way, it is much less likely that the filesystem will become full and cause DMF to abort. The appropriate size of this filesystem is

a function of the expected daily DMF transaction activity and the number of days that journals are kept.

Note: *JOURNAL_DIR* must be on a separate physical device from *HOME_DIR*.

Do not change this parameter while DMF is running.

`JOURNAL_SIZE` — Specifies the maximum size (in bytes) of the database journal file before DMF closes it and starts a new file. The default is `64000000` (or `64m`). By default, the unit of measure is bytes; see "Units of Measure" on page 275.

`LICENSE_FILE` — Specifies the full pathname of the file containing the licenses used by DMF. The default is `/etc/lk/keys.dat`. Do not change this parameter while DMF is running.

`METRICS_RETENTION` — Specifies the retention time for the DMF tape drive cumulative metrics. The cumulative metrics are reset to zero after this interval has passed since the creation of the arena object. This parameter is evaluated when a new drive object is created in the arena, typically during daemon startup. Valid values are integer followed by one of:

```
h[ours]
d[ays]
w[eeks]
```

For example, to specify five days, you could use either of the following:

```
METRICS_RETENTION 5d
METRICS_RETENTION 5days
```

By default, the cumulative metrics will be retained until the DMF daemon restarts.

Note: METRICS_RETENTION is used internally by DMF for its cumulative/averaged metrics and does not change the duration for which PCP metric archives are maintained.

NODE_BANDWIDTH

(OpenVault only) Specifies the default I/O bandwidth capacity of the node. If NODE_BANDWIDTH is not specified anywhere, the default is 1024000000000000; for more information, see "Device Block-Size Defaults and Bandwidth" on page 276. For a complete description, see "node Object" on page 294. A NODE_BANDWIDTH value specified in a node object overrides the default value specified in the base object. Also see BANDWIDTH_MULTIPLIER in "drivegroup Object Parameters" on page 397.

OV_KEY_FILE

(OpenVault only) Specifies the file containing the OpenVault security keys used by DMF. It is usually located in *HOME_DIR* and called ov_keys. There is no default. When using Parallel DMF, this file must be visible to the DMF server and all parallel data-mover nodes, therefore it must be in a CXFS filesystem. Use dmov_keyfile to create or update this file. The file should be updated if the OpenVault server name changes. Do not change this parameter while DMF is running.

OV_SERVER

(OpenVault only) Specifies the name associated with the Internet Protocol (IP) address on which the OpenVault server is listening. This should only be set if the OpenVault server is not on the same system as the DMF server. Do not change this parameter while DMF is running.

Note: More configuration steps are necessary to configure DMF to use OpenVault; see "OpenVault Configuration Tasks" on page 491.

PERFTRACE_METRICS

Enables collection of performance tracking information from DMF. Performance over time of individual components (filesystems, cartridges, tape drives, and so on) can then be graphically viewed using DMF Manager. You can set this parameter to ON or OFF. The default is OFF. If set to OFF, detailed I/O information will not be recorded by DMF and the **I/O panel** in DMF Manager cannot display certain information. See "DMF I/O" on page 248.

Note: After you change this parameter, you must restart DMF.

SERVER_NAME

Specifies the hostname of the system on which the DMF server is running. In an HA configuration, SERVER_NAME must be the HA virtual hostname rather than the output of the hostname command. This parameter is only required for HA configurations or configurations using Parallel DMF.

Note: If you change this parameter, you must copy the DMF configuration file manually to each parallel data-mover node and then restart the DMF services.

Do not change this parameter while DMF is running.

SPOOL_DIR

Specifies the base pathname for directories in which DMF log files are kept. This directory must not be in a managed filesystem. By default, this value must not reside in the root filesystem (see the description of ADMDIR_IN_ROOTFS above). The best practice is for SPOOL_DIR to be the mount point of a filesystem that is used only by DMF. In this way, it is much less likely that the filesystem will become full and cause DMF to abort. The appropriate size of this filesystem is a function of the expected daily DMF transaction activity, the MESSAGE_LEVEL parameter setting, and the number of days that logs are kept. When using Parallel DMF, *SPOOL_DIR* must either be a CXFS filesystem or

be in a CXFS filesystem. Do not change this parameter while DMF is running.

`TMP_DIR`

Specifies the base pathname for directories in which DMF puts temporary files for its own internal use. It is also used by DMF commands and scripts and is the directory used by default by the LS for caching files if the `CACHE_DIR` parameter is not defined. This directory must not be in a managed filesystem. By default, this value must not reside in the root filesystem (see the description of `ADMDIR_IN_ROOTFS` in "`base` Object Parameters" on page 278). The best practice is for *TMP_DIR* to be the mount point of a filesystem that is used only by DMF. *TMP_DIR* filesystem performance will impact the performance of many of the internal DMF administrative tasks, particularly tasks that involve the need to sort DMF databases. When using Parallel DMF, *TMP_DIR* must either be a CXFS filesystem or be in a CXFS filesystem.

Many DMF operations that do analysis on the DMF database contents use *TMP_DIR* as their work directory. Because most of these involve large buffered I/O, SGI recommends that you configure *TMP_DIR* on a fast disk, with bandwidth at the RAID level. Do not change this parameter while DMF is running.

`VALID_ROOT_HOSTS`

Specifies hostnames whose `root` users may perform certain tasks and who may also use the `dmprojid` command or equivalent `libdmfusr.so` functions.

Tasks include the following:

- Specify nondefault priority on DMF commands (such as `dmget`)
- Specify the MSPs, VGs, and MGs to which files should be migrated when using the `dmput` or `dmarchive` command, or equivalent `libdmfusr.so` functions.
- Override the order of the MSPs, VGs, and MGs from which to try to recall files on the `dmarchive`, `dmcopy`, `dmget`, `dmmove`, and `dmunput` commands,

or equivalent `libdmfusr.so` functions, overriding the default order as defined in the DMF configuration file.

The `root` user on the DMF server may always perform these actions, and therefore the DMF server name does not need to be included in `VALID_ROOT_HOSTS`. By default, only the `root` user on the DMF server may perform these actions.

Note: Customizable policies may cause this behavior to be overridden.

Changes to `VALID_ROOT_HOSTS` will take effect immediately for new requests and after no more than 10 minutes for long-running processes.

When an MSP, LS, daemon, or configuration file object (such as the `taskgroup` object named `dump_tasks` in Example 7-10, page 333) obtains a path such as *HOME_DIR* from the configuration file, the actual path used is *HOME_DIR* plus the MSP/LS/daemon object name appended as a subdirectory. For example, if `HOME_DIR` was set to `/dmf/home` in the configuration file, and the `taskgroup` object named `dump_tasks` used a value of `HOME_DIR/tapes` for the `DUMP_TAPES` parameter, then the actual path for `DUMP_TAPES` would resolve to `/dmf/home/dump_tasks/tapes`.

Note: Do not use automated space management to manage the *HOME_DIR*, *SPOOL_DIR*, or *JOURNAL_DIR* directories, because DMF daemon processes will deadlock if files that they are actively using within these directories are migrated. The `dmcheck` command reports an error if any of the *HOME_DIR*, *SPOOL_DIR*, or *JOURNAL_DIR* directories are also configured as managed filesystems. You should configure a `taskgroup` object for daemon tasks to manage old log files and journal files in these directories. See "`taskgroup` Object" on page 304 for more information.

---

base Object Examples

This section discusses the following examples:

- "`base` Object for Basic DMF" on page 287
- "`base` Object for Parallel DMF" on page 287

- "base Object for Parallel DMF in an HA Cluster" on page 288

base Object for Basic DMF

Example 7-1 base Object for Basic DMF

```
define  base
        TYPE            base
        ADMIN_EMAIL     root@dmfserver
        HOME_DIR        /dmf/home
        TMP_DIR         /dmf/tmp
        SPOOL_DIR       /dmf/spool
        JOURNAL_DIR     /dmf/journals
        JOURNAL_SIZE    10m
        OV_KEY_FILE     /dmf/home/ov_keys
enddef
```

In the above example:

- A new journal file will be created after the present file reaches 10 million bytes
- The OV_KEY_FILE parameter is necessary if OpenVault is used as the mounting service
- The OpenVault server is on the same system as the DMF server, so OV_SERVER is not specified

base Object for Parallel DMF

Example 7-2 base Object for Parallel DMF

```
define  base
        TYPE            base
        SERVER_NAME     server1
        ADMIN_EMAIL     root@dmfserver
        HOME_DIR        /dmf/home
        TMP_DIR         /dmf/tmp
        SPOOL_DIR       /dmf/spool
        JOURNAL_DIR     /dmf/journals
        JOURNAL_SIZE    10m
        OV_KEY_FILE     /dmf/home/ov_keys
enddef
```

In the above example:

- The SERVER_NAME parameter is required when using Parallel DMF. The hostname of the node that is running DMF is `server1`. OpenVault is running on the same system, so OV_SERVER is not specified.
- `/dmf/tmp` must either be a CXFS filesystem or be in a CXFS filesystem when using Parallel DMF.
- The `/dmf/spool` directory must either be a CXFS filesystem or be in a CXFS filesystem when using Parallel DMF.
- A new journal file will be created after the present file reaches 10 million bytes.
- OpenVault must be configured as the mounting service for drives that are used by parallel data-mover nodes. The `/dmf/home/ov_keys` file must be visible to the DMF server node and all parallel data-mover nodes, therefore it must be in a CXFS filesystem.

base Object for Parallel DMF in an HA Cluster

Example 7-3 base Object for Parallel DMF in an HA Cluster

```
define  base
        TYPE            base
        SERVER_NAME     virtual-server
        ADMIN_EMAIL     root@dmfserver
        HOME_DIR        /dmf/home
        TMP_DIR         /dmf/tmp
        SPOOL_DIR       /dmf/spool
        JOURNAL_DIR     /dmf/journals
        JOURNAL_SIZE    10m
        OV_KEY_FILE     /dmf/home/ov_keys
enddef
```

In the above example:

- The SERVER_NAME parameter is required when using Parallel DMF. Because this configuration is using HA, it must be set to the HA virtual hostname (in this case `virtual-server`).

Note: The INTERFACE parameter in the node objects for the DMF servers must correspond to SERVER_NAME.

- `/dmf/tmp` must either be a CXFS filesystem or be in a CXFS filesystem when using Parallel DMF.
- The `/dmf/spool` directory must either be a CXFS filesystem or be in a CXFS filesystem when using Parallel DMF.
- OpenVault must be configured as the mounting service for drives that are used by parallel data-mover nodes. The `/dmf/home/ov_keys` file must be visible to the DMF server and all parallel data-mover nodes, therefore it must be in a CXFS filesystem.
- The OpenVault server is on the same system as the DMF server, so `OV_SERVER` is not specified.

dmdaemon Object

This section discusses the following:

- "`dmdaemon` Object Name" on page 289
- "`dmdaemon` Object Summary" on page 289
- "`dmdaemon` Object Parameters" on page 290
- "`dmdaemon` Object Example" on page 293

dmdaemon Object Name

The name of the `dmdaemon` object is chosen by the administrator and may contain uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

dmdaemon Object Summary

Following is a summary of the potential parameters for the `dmdaemon` object:

```
dmdaemon
        TYPE
        EXPORT_QUEUE
        LS_NAMES  or  MSP_NAMES
        MESSAGE_LEVEL
```

```
MIGRATION_LEVEL
MOVE_FS
PARTIAL_STATE_FILES
RECALL_NOTIFICATION_RATE
TASK_GROUPS
```

dmdaemon Object Parameters

The dmdaemon object defines the configuration parameters that are necessary for operation of the DMF daemon. It is expected that you will modify the values for the pathnames and MSP names.

Parameter	Description
TYPE	Specifies dmdaemon (required name for this type of object). There is no default. **Note:** This cannot be specified as dmfdaemon. It must be dmdaemon.
EXPORT_QUEUE	Instructs the daemon to export details of its internal request queue to *SPOOL_DIR*/daemon_exports every two minutes, for use by dmstat and other utilities. On a busy system, the responsiveness of the daemon may be improved by disabling this feature. You can set this parameter to ON or OFF. The default is OFF.
LS_NAMES or MSP_NAMES	Names the LSs and MSPs used by the DMF daemon. You must specify either LS_NAMES or MSP_NAMES, but not both parameters (however, the value of either parameter can be a mixture of both LSs and MSPs). There is no default. The order of the names is significant. Where there are multiple copies of the data of migrated files, recalls will normally be directed to the first-named LS/MSP that is applicable. If more than one VG within an LS/MSP contains copies, the order of the names in the libraryserver object's DRIVE_GROUPS parameter and the drivegroup object's VOLUME_GROUPS

parameter are also significant. DMF supports up to 1024 MSPs, VGs and/or MGs.

Note: See "Ensure that the Cache Copy is Recalled First" on page 116.

Do not change these parameters while DMF is running.

`MESSAGE_LEVEL`

Specifies the highest message level that will be written to the daemon log. It must be an integer in the range `0`–`6`; the higher the number, the more messages written to the log file. The default is `2`. For more information on message levels, see Chapter 10, "Message Log Files" on page 507.

`MIGRATION_LEVEL`

Sets the highest level of migration service allowed on all DMF filesystems. From highest level of migration to least:

- `auto` (automated space management)
- `user` (requests from `dmput` or `dmmigrate` only)
- `none` (no migration)

You can also configure a lower level of migration for a specific filesystem by using the `MIGRATION_LEVEL` parameter in the `filesystem` stanza, which also permits a setting of `archive` that is used for DMF direct-archiving of unmanaged filesystems via the `dmarchive` command. See "`filesystem` Object Parameters" on page 351.

The default is `auto`.

See "`policy` Object" on page 364 for information about configuring automated space management.

`MOVE_FS`

Specifies one or more scratch filesystems that may be used when moving files between MSPs/VGs. The first filesystem named for this parameter is used as the default if the `-f` option is not specified on the `dmmove` command. This filesystem must reside in the root filesystem. Each filesystem specified must be the root of

a DMAPI-mounted filesystem (mounted with `dmi,mtpt=/`*MOVE_FS*). You must specify a value for `MOVE_FS` if you intend to use the `dmmove` command; there is no default. When using Parallel DMF, `MOVE_FS` if specified must be a CXFS filesystem.

The size of *MOVE_FS* is a function of expected `dmmove` activity. *MOVE_FS* must be mounted when a `dmmove` command is run. The best practice when using *MOVE_FS* is for it to be dedicated to the `dmmove` function. (The `dmmove` command calculates the available space in *MOVE_FS* when selecting move candidates; if other processes are allocating space in *MOVE_FS*, those calculations can become inaccurate, causing errors.)

If the scratch filesystem is managed by DMF, SGI highly recommends that you use the `-s` option when executing the `dmmove` or `dmemptytape` commands to prevent the filesystem free-space minimum from being reached as a result of the command's execution. For more information, see the `dmmove`(8) and `dmemptytape`(8) man pages.

`PARTIAL_STATE_FILES` Enables or disables the DMF daemon's ability to produce partial-state files. The possible values are:

- `ON`, which means that the daemon will correctly process `put` and `get` requests that would result in a partial-state file. The default is `ON`.
- `OFF`, which means that all `put` and `get` requests that require a change to the online status of the file will result in a file that is completely online or offline. That is, any `put` request that makes any part of the file offline will result in the entire file being made offline. Any `get` request that would result in any part of the file being brought back online will result in the entire file being brought back online.

`RECALL_NOTIFICATION_RATE` Specifies the approximate rate (in seconds) at which regions of a file being recalled are put online. This allows for access to part of a file before the entire file is

recalled. The default is 30 seconds. Specify a value of `0` if you want the user process to be blocked until the entire recall is complete. The optimum setting of this parameter is dependent on many factors and must be determined by trial and error. The actual rate at which regions being recalled are put online may vary from the value of `RECALL_NOTIFICATION_RATE`.

`TASK_GROUPS` — Names the `taskgroup` objects that contain tasks the daemon should run. By default, no tasks are run. For more information, see "`taskgroup` Object" on page 304. SGI recommends that you use the task groups specified in the sample configuration files, changing the parameters as necessary for your site.

dmdaemon Object Example

Example 7-4 `dmdaemon` object

```
define  daemon
        TYPE                    dmdaemon
        MOVE_FS                 /dmmove_dir
        LS_NAMES                lib1 ftp2
        TASK_GROUPS             daemon_tasks dump_tasks
enddef
```

In the above example:

- The name of the `dmdaemon` object is `daemon`.
- The `dmmove` command will use the `/dmmove_dir` filesystem as a scratch filesystem.
- The names of the LSs are `lib1` and `ftp2`.
- The daemon will run the tasks specified by the `daemon_tasks` and `dump_tasks` objects (see Example 7-13, page 336 and Example 7-10, page 333). In the example, `daemon_tasks` defines the tasks such as scanning and managing log files and journal files. The `dump_tasks` object defines tasks that back up managed filesystems.
- The `MIGRATION_LEVEL` level is not explicitly set, so the default of `auto` is used.

node Object

- "node Object Name" on page 294
- "node Object Summary" on page 294
- "node Object Parameters" on page 294
- "node Object Examples" on page 297

node Object Name

The name of the node object must be the same as the output of the hostname command.

node Object Summary

Following is a summary of the potential parameters for the node object:

```
node
        TYPE
        HBA_BANDWIDTH
        INTERFACE
        MERGE_INTERFACE
        NODE_BANDWIDTH
        SERVICES
```

node Object Parameters

Note: The node object is only for Parallel DMF configurations. Basic DMF configurations do not use the node object.

The name of the node object must match the name returned by hostname on the system. In a Parallel DMF configuration, there must be a node object for the DMF server and every parallel data-mover node. In a Parallel DMF configuration in an HA cluster, the node objects for the DMF servers will have identical parameter values.

Parameter	Description
TYPE	Specifies node (required name for this type of object). There is no default.
HBA_BANDWIDTH	*(OpenVault only)* Specifies the I/O bandwidth capacity of an HBA port that is connected to drives on the node. The value is in bytes per second. All of the HBA ports connected to drives on a node are assumed to have the same bandwidth capacity. The LS uses this value when determining which drives to use. The maximum is 1024000000000000. The minimum is 0, which means that the HBA will not be used. HBA_BANDWIDTH can also be specified in the base object, in which case the value there is used as the default (values specified in a node object override the default for that node); see "base Object Parameters" on page 278. If HBA_BANDWIDTH is not specified in the base object, the default is 1024000000000000. For more information, see "Device Block-Size Defaults and Bandwidth" on page 276. Also see BANDWIDTH_MULTIPLIER in "drivegroup Object Parameters" on page 397.
INTERFACE	Specifies the IP address or associated name of this node to be used for communication between DMF components. By default, the system hostname will be used. If this node is a potential DMF server in an HA configuration, this parameter must match the virtual hostname used for SERVER_NAME in the base object. See "Use a Private Network Interface in a Parallel Environment" on page 121.
MERGE_INTERFACE	Specifies the IP address or associated name on this node to be used when merging sparse volumes via sockets. The default is to use the same interface used for other DMF communication (see INTERFACE above). See "Use a Private Network Interface in a Parallel Environment" on page 121.
NODE_BANDWIDTH	*(OpenVault only)* Specifies the I/O bandwidth capacity of this node, in bytes per second. The LS uses this value to calculate how many drives it can

simultaneously use on a node. The maximum is 1024000000000000. The minimum is 0, which means that the node will not be used.

NODE_BANDWIDTH can also be specified in the base object, in which case the value there is used as the default (values specified in a node object override the default for that node); see "base Object" on page 277. If NODE_BANDWIDTH is not specified in the base object, the default is 1024000000000000. For more information, see "Device Block-Size Defaults and Bandwidth" on page 276. Also see BANDWIDTH_MULTIPLIER in "drivegroup Object Parameters" on page 397.

SERVICES — Specifies the name of the services object used to configure DMF services on this node. Multiple nodes may refer to the same services object. For node-specific configuration, each node can refer to a different services object. If no SERVICES parameter is defined, the default values for the services object parameters are used.

node Object Examples

This section discusses the following examples:

- "node Objects for Parallel DMF" on page 297
- "node Objects for Parallel DMF in an HA Cluster" on page 298

node Objects for Parallel DMF

Example 7-5 node Objects for Parallel DMF

```
define server1
        TYPE                    node
        INTERFACE               server1-dmfnet
        SERVICES                server1_services
enddef

define pdm1
        TYPE                    node
        INTERFACE               pdm1-dmfnet
        SERVICES                pdm1_services
enddef
```

In the above example:

- There are two data movers: the DMF server `server1` and the parallel data-mover node `pdm1`.
- The DMF services on the `server1` node use the parameters defined in the `server1_services` object. The DMF services on the `pdm1` node use the parameters defined in the `pdm1_services` object.
- Because `INTERFACE` is defined, the nodes will communicate on the IP addresses associated with the hostnames `server1-dmfnet` and `pdm1-dmfnet`. (If `INTERFACE` was not defined, they would communicate using `server1` and `pdm1`.)

node Objects for Parallel DMF in an HA Cluster

Example 7-6 node Objects for Parallel DMF in an HA Cluster

```
define server1
        TYPE                    node
        INTERFACE               virtual-server
        SERVICES                dmfserver_services
enddef

define server2
        TYPE                    node
        INTERFACE               virtual-server
        SERVICES                dmfserver_services
enddef

define pdm1
        TYPE                    node
        SERVICES                pdm1_services
enddef
```

In the above example:

- The following nodes are data movers:
  - Either the potential DMF server `server1` or the potential DMF server `server2` (for example, `server1` could be the active DMF server and `server2` could be the passive DMF server)
  - The parallel data-mover node `pdm1`

Note: At any given time, only one of the potential DMF server nodes (either `server1` or `server2`) may provide data mover functionality.

- The virtual hostname in the HA cluster is `virtual-server`. The `INTERFACE` parameter is required in an HA cluster for the potential DMF servers and it must match the value for `SERVER_NAME` in the `base` object (see "`base` Object" on page 277).
- The potential DMF server nodes provide the tasks that are described by the `dmfserver_services` object. The parallel data-mover node provides the DMF services described by the `pdm1_services` object.

- Because `server1` and `server2` are both potential servers in this HA configuration, they should specify identical parameters and parameter values.
- The nodes will communicate using the IP addresses associated with the `virtual-server` and `pdm1` hostnames.

`services` Object

This section discusses the following:

- "`services` Object Name" on page 299
- "`services` Object Summary" on page 299
- "`services` Object Parameters" on page 300
- "`services` Object Examples" on page 301

`services` Object Name

The name of the `services` object is chosen by the administrator and may contain uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

`services` Object Summary

Following is a summary of the potential parameters for the `services` object:

```
services
        TYPE
        MESSAGE_LEVEL
        NODE_ANNOUNCE_RATE
        NODE_TIMEOUT
        SERVICES_PORT
        TASK_GROUPS
```

services Object Parameters

The services object defines parameters for dmnode_service and other DMF services. When using Parallel DMF, multiple services objects may be defined. For basic DMF configurations, exactly one services object may be defined. (The services parameters all have defaults, so you only need to define a services object if you want to change those defaults.)

Parameter	Description
TYPE	Specifies services (required name for this type of object). If you include this object, you must specify this parameter.
MESSAGE_LEVEL	Specifies the highest message level that will be written to the service logs. It must be an integer in the range 0–6; the higher the number, the more messages written to the log file. The default is 2. For more information on message levels, see Chapter 10, "Message Log Files" on page 507.
NODE_ANNOUNCE_RATE	Specifies the rate (in seconds) at which the DMF server or parallel data-mover node will contact the dmnode_service on the DMF server to announce its presence. This also determines the rate at which configuration changes are propagated to any parallel data-mover nodes. This value should be less than the value of NODE_TIMEOUT. The default is 20 seconds.
NODE_TIMEOUT	Specifies the number of seconds after which the data mover functionality on the DMF server or on a parallel data-mover node will be considered inactive if it has not contacted the dmnode_service on the DMF server. This value should be larger than the value of NODE_ANNOUNCE_RATE. The default is 180 seconds.
SERVICES_PORT	Specifies the port number on which DMF starts a locator service, which DMF uses to locate other DMF services. It must be an integer in the range 1–65535. The default is 11108.

Note: If you change this parameter, you must copy the DMF configuration file manually to each parallel data-mover node and then restart the DMF services.

Do not change this parameter while DMF is running.

TASK_GROUPS — Names the `taskgroup` objects that contain scripts to be run on the DMF server and every parallel data-mover node. (This is unlike the TASK_GROUPS parameters of other objects, which contain scripts to be run on just the DMF server.) If you specify this parameter, you must specify the scripts to be run. For more information, see "`taskgroup` Object" on page 304. By default, no tasks are run. SGI recommends that you use the task groups specified in the sample configuration files, changing the parameters as necessary for your site.

`services` Object Examples

This section discusses the following examples:

- "`services` object for Parallel DMF" on page 301
- "`services` Object for Parallel DMF in an HA Cluster" on page 303

`services` object for Parallel DMF

Example 7-7 `services` object for Parallel DMF

```
define server1_services
        TYPE                    services
        MESSAGE_LEVEL           2
        TASK_GROUPS             node_tasks
enddef

define pdm1_services
        TYPE                    services
        MESSAGE_LEVEL           4
        SERVICES_PORT           1111
        TASK_GROUPS             node_tasks
```

```
enddef
```

In the above example:

- Two services are defined:
  - `server1_services` (which applies to `server1`, as shown in Example 7-5, page 297)
  - `pdm1_services` (which applies to `pdm1`, as also shown in Example 7-5)
- The `server1` services will log fewer messages than the `pdm1` services.
- The `pdm1` services use locator port `1111`. The `server1` services will use the default port.
- Both services use the tasks described by the `node_tasks` object.

`services` Object for Parallel DMF in an HA Cluster

Example 7-8 `services` Object for Parallel DMF in an HA Cluster

```
define dmfserver_services
        TYPE                    services
        MESSAGE_LEVEL           2
        TASK_GROUPS             servernode_tasks
enddef

define pdm1_services
        TYPE                    services
        MESSAGE_LEVEL           4
        SERVICES_PORT           1111
        TASK_GROUPS             movernode_tasks
enddef
```

In the above example:

- Two services are defined:
  - `dmfserver_services`, which applies to `server1` and `server2` (as shown in Example 7-6, page 298)
  - `pdm1_services`, which applies to `pdm1` (as shown in Example 7-6)
- The `dmfserver_services` services will log fewer messages than the `pdm1` services.
- The `pdm1` services use locator port `1111`. The `dmfserver_services` services will use the default port.
- The active DMF server (either `server1` or `server2`) will run the tasks defined by the `servernode_tasks` object.
- The parallel data-mover node `pdm1` will run the tasks defined by the `movernode_tasks` object.

`taskgroup` Object

This section discusses the following:

- "Overview of the Tasks" on page 304
- "Backup Tasks" on page 309
- "`taskgroup` Object Name" on page 310
- "`taskgroup` Object Summary" on page 310
- "`taskgroup` Object Parameters" on page 312
- "`taskgroup` Object Examples" on page 332

See also "Precedence of Objects for Integrated Backups" on page 154.

Overview of the Tasks

You can configure `taskgroup` objects to manage how periodic maintenance tasks are performed. The object that performs the tasks refers to the `taskgroup` name in its stanza. You can configure when each task should run. For some of the tasks, you must provide more information. Table 7-1 summarizes the tasks.

Table 7-1 Automated Maintenance Task Summary

Referencing Object Type	Task	Purpose	Associated Parameters
dmdaemon	run_audit.sh	Audit databases	
	run_copy_databases.sh	Back up DMF databases	DATABASE_COPIES
	run_daily_drive_report.sh	Create a report about physical tape drives that have indicated they need cleaning	DRIVETAB
	run_daily_report.sh[1]	Create a report including information on managed filesystems (if run_filesystem_scan.sh has been run recently) and DCM MSPs, and all LSs	
	run_daily_tsreport.sh	Create a report containing the output of the tsreport command, which reports physical tape drive alerts, errors, and statistics	DRIVETAB TSREPORT_OPTIONS
	run_dmmigrate.sh	Run dmmigrate on all filesystems defined with automated space-management or user space-management enabled	DMMIGRATE_MINIMUM_AGE DMMIGRATE_TRICKLE DMMIGRATE_UNACK DMMIGRATE_VERBOSE DMMIGRATE_WAIT

1 The run_compact_tape_report.sh and run_tape_report.sh tasks have been superseded by the run_daily_report.sh task.

Referencing Object Type	Task	Purpose	Associated Parameters
	run_filesystem_scan.sh	Run dmscanfs on filesystems to collect file information for subsequent use by other scripts and programs	SCAN_FILESYSTEMS SCAN_FOR_DMSTAT SCAN_OUTPUT SCAN_PARALLEL SCAN_PARAMS
	run_full_dump.sh	*(disk, tape, and integrated backups only)* Full backup of filesystems [2]	See Table 7-2 on page 309
	run_hard_deletes.sh	Hard-delete files that are no longer on backup media	Uses DUMP_RETENTION
	run_partial_dump.sh	*(disk, tape, and integrated backups)* Perform an incremental backup of filesystems	Associated with the same parameters as run_full_dump.sh plus UPDATE_CTIME. (If DUMP_MAX_FILESPACE is set but UPDATE_CTIME is not, run_partial_dump.sh will perform a full backup rather than an incremental backup for managed filesystems. See "DUMP_MAX_FILESPACE Requires UPDATE_CTIME for Incremental Backups" on page 148.)
	run_remove_alerts.sh	Remove old alert records	ALERT_RETENTION MAX_ALERTDB_SIZE REMALERT_PARAMS
	run_remove_journals.sh	Remove old journal files	JOURNAL_RETENTION
	run_remove_logs.sh	Remove old log files	LOG_RETENTION
	run_remove_perf.sh	Remove old performance records	MAX_PERFDB_SIZE PERF_RETENTION REMPERF_PARAMS

2 For restores, see the dmxfsrestore(8) man page.

Referencing Object Type	Task	Purpose	Associated Parameters
	run_scan_logs.sh	Scan recent log files for errors	
drivegroup	run_merge_mgr.sh	Merge sparse volumes	DATA_LIMIT THRESHOLD VOLUME_LIMIT
libraryserver	run_merge_stop.sh	Stop volume merges for only those volumes that were previously marked as sparse by either the run_tape_merge.sh or run_merge_mgr.sh scripts.	
	run_fmc_free.sh	Free the volumes in the fast-mount cache that meet the criteria	FILE_RETENTION_DAYS FMC_MOVEFS FREE_VOLUME_MINIMUM FREE_VOLUME_TARGET
	run_tape_merge.sh	Merge sparse volumes	DATA_LIMIT THRESHOLD VOLUME_LIMIT
DCM msp	run_dcm_admin.sh	Routine DCM MSP administration	
	run_dmmigrate.sh	Run dmmigrate on the specified *STORE_DIRECTORY* (for this DCM MSP only)	DMMIGRATE_MINIMUM_AGE DMMIGRATE_TRICKLE DMMIGRATE_UNACK DMMIGRATE_VERBOSE DMMIGRATE_WAIT
filesystem	run_dmmigrate.sh	Run dmmigrate on the specified filesystem if it is defined with a MIGRATION_LEVEL of auto or user	DMMIGRATE_MINIMUM_AGE DMMIGRATE_TRICKLE DMMIGRATE_UNACK DMMIGRATE_VERBOSE DMMIGRATE_WAIT

Referencing Object Type	Task	Purpose	Associated Parameters
services	run_remove_logs.sh	Remove old log files	LOG_RETENTION
volumegroup	run_fmc_free.sh	Free the volumes in the fast-mount cache that meet the criteria	FILE_RETENTION_DAYS FMC_MOVEFS FREE_VOLUME_MINIMUM FREE_VOLUME_TARGET

Backup Tasks

You can schedule backups to be performed via `xfsdump` in the DMF configuration file. When defining a backup task, you must provide information such as:

- Tape and device names
- Retention times for output
- Whether to migrate files before backing up the filesystem
- Locations for inventory files

Not all tasks and parameters apply to each backup method. Table 7-2 lists backup parameters according to method.

Table 7-2 Dump Parameters According to DMF Backup Method

DMF Backup Method	Parameters

Disk, tape, and integrated backups	`DUMP_FILE_SYSTEMS`
	`DUMP_FLUSH_DCM_FIRST`
	`DUMP_MIGRATE_FIRST`
	`DUMP_RETENTION`
	`DUMP_MAX_FILESPACE`
	`DUMP_XFSDUMP_PARAMS`

Disk and integrated backups only	`DUMP_COMPRESS`
	`DUMP_CONCURRENCY`
	`DUMP_DESTINATION`
	`DUMP_MIRRORS`
	`DUMP_STREAMS`
	`DUMP_WORKSPACE`

DMF Backup Method	Parameters
Integrated backups only	DUMP_DR_EMAIL DUMP_DR_EMAIL_MAX_SIZE DUMP_DR_MINIMAL DUMP_DR_MIRRORS DUMP_GROUPS DUMP_MISC_LOCATIONS DUMP_POST_BACKUP_SCRIPT DUMP_POST_DR_SCRIPT DUMP_PRE_BACKUP_SCRIPT
Tape backups only	DUMP_DEVICE DUMP_INVENTORY_COPY DUMP_TAPES DUMP_VSNS_USED

For more information, see Chapter 4, "Backups and DMF" on page 141.

taskgroup Object Name

The name of the taskgroup object is chosen by the administrator and may contain uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

taskgroup Object Summary

Following is a summary of the potential parameters for the taskgroup object:

```
taskgroup
        TYPE
        ALERT_RETENTION
        DATA_LIMIT
        DATABASE_COPIES
        DMMIGRATE_MINIMUM_AGE
        DMMIGRATE_MIN_OFFLINE_AGE
        DMMIGRATE_MIN_OFFLINE_SIZE
        DMMIGRATE_QUEUE_MAX
        DMMIGRATE_TRICKLE
        DMMIGRATE_UNACK
        DMMIGRATE_VERBOSE
```

```
DMMIGRATE_WAIT
DRIVETAB
DUMP_COMPRESS
DUMP_CONCURRENCY
DUMP_DESTINATION
DUMP_DEVICE
DUMP_DR_EMAIL
DUMP_DR_EMAIL_MAX_SIZE
DUMP_DR_MINIMAL
DUMP_DR_MIRRORS
DUMP_FILE_SYSTEMS
DUMP_FLUSH_DCM_FIRST
DUMP_GROUPS
DUMP_INVENTORY_COPY
DUMP_MAX_FILESPACE
DUMP_MIGRATE_FIRST
DUMP_MIRRORS
DUMP_MISC_LOCATIONS
DUMP_POST_BACKUP_SCRIPT
DUMP_POST_DR_SCRIPT
DUMP_PRE_BACKUP_SCRIPT
DUMP_RETENTION
DUMP_STREAMS
DUMP_TAPES
DUMP_VSNS_USED
DUMP_WORKSPACE
DUMP_XFSDUMP_PARAMS
FILE_RETENTION_DAYS
FMC_MOVEFS
FMC_NAME
FREE_VOLUME_MINIMUM
FREE_VOLUME_TARGET
JOURNAL_RETENTION
LOG_RETENTION
MAX_ALERTDB_SIZE
MAX_PERFDB_SIZE
PERF_RETENTION
REMALERT_PARAMS
REMPERF_PARAMS
RUN_TASK
SCAN_FILESYSTEMS
```

```
SCAN_FOR_DMSTAT
SCAN_OUTPUT
SCAN_PARALLEL
SCAN_PARAMS
THRESHOLD
TSREPORT_OPTIONS
VOLUME_LIMIT
```

taskgroup Object Parameters

Note: For integrated backups, you can override many of the dump parameters for a particular filesystem by setting the parameters in other objects, as noted below. See:

- "Precedence of Objects for Integrated Backups" on page 154
- "dumpgroup Object Parameters" on page 343
- "filesystem Object Parameters" on page 351
- "msp Object Parameters for a DCM MSP" on page 460.

The taskgroup object parameters are as follows:

TYPE — Specifies taskgroup (required name for this type of object). There is no default.

ALERT_RETENTION — Specifies the age of alert records that are preferred to be kept when the run_remove_alerts.sh task is run. The run_remove_alerts.sh task uses but does not require this parameter. This parameter is evaluated only when run_remove_alerts.sh is run. Valid values are an integer followed by one of:

m[inutes]
h[ours]
d[ays]
w[eeks]

Note: The set of records chosen for deletion is the union of the records that are older than the ALERT_RETENTION value and the oldest records when the database reaches the MAX_ALERTDB_SIZE threshold size:

- If you specify ALERT_RETENTION without MAX_ALERTDB_SIZE, older records will be deleted no matter how small the database is
- If you specify MAX_ALERTDB_SIZE without ALERT_RETENTION, records will be kept no matter how old they are, so long as the database remains below the threshold size
- If you specify neither parameter, no records will be deleted (and REMALERT_PARAMS is ignored)

DATA_LIMIT — Specifies the maximum amount of data (in bytes) that should be selected for merging at one time. By default, there is no limit.

DATABASE_COPIES — Specifies one or more directories into which the run_copy_databases.sh task will place a copy of the DMF databases. The run_copy_databases.sh task copies a snapshot of the current DMF databases to the directory with the oldest copy. If you specify multiple directories, you should spread the directories among multiple disk devices in order to minimize the chance of losing all copies of the databases. There is no default. This directory must not be in a managed filesystem.

DMMIGRATE_MINIMUM_AGE — Specifies the value used for the -m option (in minutes) when dmmigrate is executed by the run_dmmigrate.sh task. The default is 10. For more information, see the dmmigrate(8) man page.

DMMIGRATE_MIN_OFFLINE_AGE — Specifies the value used for the -p option (in minutes) when dmmigrate is executed by the run_dmmigrate.sh task. This value is a file's minimum age in order for its data blocks to be released, making the file offline. This value must be a positive

integer equal to or greater than the value of DMMIGRATE_MINIMUM_AGE. The default is to never release the data blocks. For more information, see the dmmigrate(8) man page.

DMMIGRATE_MIN_OFFLINE_SIZE — Specifies the value used for the -s option (in bytes) when dmmigrate is executed by the run_dmmigrate.sh task. This value is a file's minimum size in order for its data blocks to be released, making the file offline. By default, there is no size limit. The value may have a trailing scaling character. For more information, see the dmmigrate(8) man page.

DMMIGRATE_QUEUE_MAX — Specifies the maximum number of dmmigrate requests that can be outstanding in the DMF daemon queue at any given time when dmmigrate is executed by the run_dmmigrate.sh task; when this threshold is reached, the command waits before issuing more requests. By default, the queue may contain 2 million dmmigrate requests. The minimum is the DMMIGRATE_UNACK value (if specified), or else 1; the maximum is 2 billion (2000000000).

DMMIGRATE_TRICKLE — Specifies the value used for the -t option when dmmigrate is executed by the run_dmmigrate.sh task.. You can set this parameter to ON or OFF. The default is ON. For more information, see the dmmigrate(8) man page.

DMMIGRATE_UNACK — Specifies the value used for the -r option when dmmigrate is executed by the run_dmmigrate.sh task. Valid values are 1—200000, the default is 10. This parameter is valid only when DMMIGRATE_TRICKLE is set to ON. For more information, see the dmmigrate(8) man page.

DMMIGRATE_VERBOSE — Specifies the value used for the -v option when dmmigrate is executed by the run_dmmigrate.sh task. You can set this parameter to ON or OFF. The default is OFF. For more information, see the dmmigrate(8) man page.

DMMIGRATE_WAIT — Specifies the value used for the -w option when dmmigrate is executed by the run_dmmigrate.sh

task. You can set this parameter to ON or OFF. The default is OFF. For more information, see the dmmigrate(8) man page.

DRIVETAB

Provides the name of a file that is used with the tsreport --drivetab option, which causes the run_daily_drive_report and run_daily_tsreport output to contain the drive name instead of the device name, making the report more readable. By default, the device name is reported.

DUMP_COMPRESS

(disk and integrated backups only) Specifies the compression type and level to be used when dumping the filesystems. The following values are accepted:

OFF	No compression.
ON	Equivalent to gzip:1 (default).
gzip[:*level*]	Use gzip(1) with the specified compression level. If you do not specify *level*, a value of 1 is used.
bzip2[:*level*]	Use bzip2(1) with the specified compression level. If you do not specify *level*, a value of 9 is used.

For more information about legal values for *level*, see the man page for the compression tool.

For example, for a compression level of 3 with bzip2, you would use the following:

```
DUMP_COMPRESS       bzip2:3
```

Note: On backups consisting largely of migrated files, gzip:1 (or ON) gives by far the best performance without sacrificing compression.

For integrated backups, the value set in a dumpgroup object or filesystem object overrides the value set here (the dumpgroup object has the highest precedence).

DUMP_CONCURRENCY — *(disk and integrated backups only)* Specifies the maximum number of filesystems that will be backed up simultaneously. By default, there is no limit.

DUMP_DESTINATION — *(disk and integrated backups only)* Specifies the directory in which to store backups, according to backup method:

- **Disk backup:** this directory must not be in a managed filesystem. If the filesystem is listed in /etc/fstab and is not mounted when backups or restores are started (using the noauto mount option), it will be mounted automatically for the duration of the operation; if the filesystem is on a COPAN RAID set, it must be a local filesystem and it will consume 1 unit from the power budget whenever it is mounted.

  To ensure optimal dump performance for a disk backup, SGI recommends that the DUMP_DESTINATION directory is not in an NFS filesystem. However, if an NFS filesystem is required, you should set the DUMP_WORKSPACE parameter to a local filesystem that can hold the temporary files created during the dump process.

- **Integrated backups:** this directory must be the top-level directory (that is, not subdirectory) of a managed filesystem that is already mounted, and USE_AS_DUMP_DEST must be set to ON (see "filesystem Object Parameters" on page 351).

  Note: Integrated backups are not normally used with COPAN massive array of idle disks (MAID). However, if the filesystem is on a COPAN RAID set, it must be a constantly mounted managed filesystem and it will consume 1 unit from the power budget.

DUMP_DEVICE — *(tape backups only)* Specifies the name of the `drivegroup` or `device` object in the configuration file that defines how to mount the tapes that the backup tasks will use.

DUMP_DR_EMAIL — *(integrated backups only)* Specifies one or more email addresses to which the disaster-recovery (DR) bundle is sent.

DUMP_DR_EMAIL_MAX_SIZE — *(integrated backups only)* Specifies the maximum size of the DR bundle that will be attached to the email sent to the addresses specified by `DUMP_DR_EMAIL`. By default, there is no limit on the size of the bundle.

DUMP_DR_MINIMAL — *(integrated backups only)* Specifies whether or not the backup process produces a DR bundle of the smallest possible size. When set to `ON`, the bundle will contain only a dump of the `DUMP_DESTINATION` filesystem and the filesystem's metadata. When set to `OFF` (the default), the bundle will also include the configuration details and the XFS inventories. You can set this parameter to `ON` or `OFF`. The default is `OFF`.

DUMP_DR_MIRRORS — *(integrated backups only)* Specifies one or more directories in which to place a copy of the DR bundle. This should be a remote filesystem accessible through passwordless `ssh`. After the initial copy is made to the `DUMP_DESTINATION` directory, a copy will be made in each of the disaster-recovery mirrors, specified with the following syntax:

[*user@*]*hostname*:*directoryname*

Note: For data security, SGI recommends that you do not specify a local directory.

For example:

```
admin@myhost:/usr/mirrorA
```

DMF transfers files by using SSH. You must set up SSH keys so that the local `root` user can log in to the remote host as a remote user without a password.

See the ssh-keygen(1)and ssh-copy-id(1) man pages for details. This parameter is required when using integrated backups.

DUMP_FILE_SYSTEMS — *(disk, tape, and integrated backups)* Specifies one or more filesystems to back up, according to the backup method:

- **Disk/tape backups:** Specifies one or more filesystems to back up. By default, the tasks will back up all of the managed filesystems configured in the configuration file. Use this parameter only if your site needs different backup policies (such as different backup times) for specific filesystems or wishes to back up filesystems that are not managed by DMF. It is safest not to specify this parameter and therefore back up all filesystems configured for management by DMF.
- **Integrated backups:** By default, the task will back up all of the managed filesystems. Use this parameter only if your site needs different backup policies (such as different compression type) for different filesystems or wishes to back up filesystems that are not managed by DMF. It is safest to exclude this parameter and therefore back up all filesystems configured for management by DMF. If DUMP_GROUPS is configured but DUMP_FILE_SYSTEMS is not, only those filesystems defined by the DUMP_FILE_SYSTEMS parameters in the specified dumpgroup objects will be dumped; see "dumpgroup Object" on page 342.

For integrated backups, the value set in a dumpgroup object or filesystem object overrides the value set here (the dumpgroup object has the highest precedence).

DUMP_FLUSH_DCM_FIRST — *(disk, tape, and integrated backups)* Specifies whether or not the dmmigrate command is run before dumping the filesystems specified. Running dmmigrate first ensures that all non-dual-resident files in the DCM MSP caches are migrated to a lower tier. If DUMP_MIGRATE_FIRST is also enabled, it is processed

first. You can set this parameter to ON or OFF. The default is ON. For integrated backups, the value set in an msp object for a DCM MSP overrides the value set here.

DUMP_GROUPS

(integrated backups only) Specifies a list of dumpgroup objects separated by spaces. Each dumpgroup specifies one or more filesystems that should be dumped with specific characteristics that override the settings in this taskgroup object for any of the following parameters:

DUMP_COMPRESS
DUMP_MAX_FILESPACE
DUMP_MIGRATE_FIRST
DUMP_STREAMS
DUMP_XFSDUMP_PARAMS

See:

- "Precedence of Objects for Integrated Backups" on page 154
- "dumpgroup Object Parameters" on page 343

DUMP_INVENTORY_COPY

(tape backups only) Specifies the pathnames of one or more directories into which are copied the XFS inventory files for the backed-up filesystems. If you specify multiple directories, spreading the directories among multiple tape devices minimizes the chance of losing all copies of the inventory. The backup scripts choose the directory with the oldest inventory copy and copy the current one to it.

Note: For disk backup, copies of the inventory are maintained in the directory specified by DUMP_DESTINATION.

DUMP_MAX_FILESPACE

(disk, tape, and integrated backups) Specifies the maximum size of regular files to be included in a dump. (This limit does not apply to offline, dual-state, and partial-state files.) Regular files using more than this space are silently left out. This value applies to all filesystems being dumped except for the DMF

databases. The value is in bytes. You may also specify multiples of KB, MB, or GB by specifying a suffix of `k`, `m`, or `g`, respectively; however, DMF will round up the value to 1 KB if you provide a value that equates to less than 1 KB. If this parameter is not provided, there is no limit.

For integrated backups, the value set in a `dumpgroup` object or `filesystem` object overrides the value set here (the `dumpgroup` object has the highest precedence). **This parameter can be dangerous** because files are omitted silently and the limit applies to all of the filesystems specified by this object (which by default is all filesystems). Therefore SGI recommends that you set this limit only for specific filesystems.

When this parameter is defined, you must set the `UPDATE_CTIME` parameter to `ON` in the `filesystem` object if you want to use incremental dumps on managed filesystems; see "`filesystem` Object Parameters" on page 351.

`DUMP_MIGRATE_FIRST` *(disk, tape, and integrated backups)* Specifies whether or not the `dmmigrate` command is run before dumping the filesystems. Running `dmmigrate` first ensures that all migratable files in the managed filesystems are migrated, thus reducing the amount of backup media space and making it run much faster. You can set this parameter to `ON` or `OFF`. The default is `ON`. For integrated backups, the value set in a `dumpgroup` object or `filesystem` object overrides the value set here (the `dumpgroup` object has the highest precedence).

`DUMP_MIRRORS` *(disk and integrated backups only)* Specifies one or more directories in which to place a copy of the dumps. After the initial copy is made to the `DUMP_DESTINATION` directory, a copy will be made in each of the mirrors, which may be local or remote:

- A **local** directory is specified by beginning with a '`/`' character. If the filesystem containing this directory is listed in `/etc/fstab` and is not mounted when

mirroring begins, it will be mounted automatically for the duration of the mirror operation.

- A **remote** directory is specified with the following syntax:

  [*user*@]*host*:*path*

  For example:

  admin@myhost:/usr/mirrorA

DMF transfers files by using SSH. You must set up SSH keys so that the local root user can log in to the remote host as a remote user without a password. See the ssh-keygen(1) and ssh-copy-id(1) man pages for details.

If DUMP_DESTINATION is a managed filesystem configured with USE_AS_DUMP_DEST set to ON, the user is responsible for managing space in DUMP_MIRRORS.

DUMP_MISC_LOCATIONS — *(integrated backups only)* Specifies one or more full pathnames that must be included in the dump. The dumps of these files and directories are archived under the *DUMP_DESTINATION*/misc directory are migrated to volumes along with other dumps, but are not part of the DR bundle. (The *DUMP_DESTINATION*/misc directory and its contents are migrated like other files in the backup filesystem. The procedure to retrieve this directory is the same as the procedure to retrieve other filesystem dumps and DMF databases. See "Disaster Recovery Procedures" on page 650.)

DUMP_POST_BACKUP_SCRIPT — *(integrated backups only)* Specifies the full pathname to a user script that is run after the filesystem dumps, database snapshots, and so on are complete but just before the DR bundle is created. Hard links to files created by the dump process are placed under *DUMP_DESTINATION*/current. For example, you can use this script to store these files in a remote location, to place other files in the

DUMP_DESTINATION filesystem, or to prune older files based on a custom dump-retention policy. However, you must manage the pruning of files that you explicitly add to the *DUMP_DESTINATION* filesystem if those files are not included and managed by the dump process. Normally the dump sessions prune files in the following directories that are older than *DUMP_RETENTION*:

```
configs
databases
filesystems
inventories
misc
```

If there are additional files or directories that are placed in *DUMP_DESTINATION*, they are migrated during the dump session but are not pruned during the normal pruning process.

`DUMP_POST_DR_SCRIPT` *(integrated backups only)* Specifies the full pathname to a user script that is run just after the DR bundle is created. For example, you could use this script to transfer the DR bundle to one or more safe locations. The dump script calls *DUMP_POST_DR_SCRIPT* with just one argument, which is the temporary location of the DR bundle. You can write your script accordingly if you want to put the DR bundle in a safe place using some mechanism other than `ssh` (such as using `ftp`).

`DUMP_PRE_BACKUP_SCRIPT` *(integrated backups only)* Specifies the full pathname to a user script that is run just before the backup is performed. For example, you can use this script to store user or administrative files in one of the dumped filesystems before the backup session begins.

DUMP_RETENTION

(disk, tape, and integrated backups) Specifies how long the filesystem backups will be kept before the tape or disk space is reused. This parameter is evaluated as follows:

- For tape backups, this parameter it is evaluated after a successful **full** backup (but not after incremental backups). Also, if at the start of the backup process it is found that no tape is available, a prune is attempted in order to free tapes that are holding data older than the specified DUMP_RETENTION period.
- For disk backups and integrated backups, this parameter is evaluated at the end of every full backup, every incremental backup, and every prune-only backup (a prune is attempted at the end of every partial or full backup).

You can specify this value as follows:

- As a single value, in which case all backups older than the value will be removed (supported for all methods)
- As a pair of minimum and maximum values (separated by a space), in which case backups will be kept for the minimum age and up to the maximum age as space permits (supported for disk backup only)

The run_hard_deletes.sh, run_partial_dump.sh, and run_full_dump.sh tasks require this parameter.

Valid values are an integer followed by one of:

m[inutes]
h[ours]
d[ays]
w[eeks]

DUMP_STREAMS

(disk and integrated backups only) Specifies the number of xfsdump streams (threads) to use when dumping a filesystem. Using multiple streams can reduce backup and restore times. The default is 1 and the maximum is 20. For integrated backups, the value set in a

dumpgroup object or filesystem object overrides the value set here (the dumpgroup object has the highest precedence).

DUMP_TAPES

(tape backups only) Specifies the path of a file that contains tape volume serial numbers (VSNs), one per line, for the backup tasks to use. A VSN line in the specified file that begins with a comment character (#) is considered to be a temporarily disabled VSN that is unavailable for backups but whose xfsdump inventory records should be preserved if they exist. Any other text in the file after a comment character is considered to be a comment. For example, the file could contain the following:

```
VSN001
VSN002  # a comment

# the following VSN is temporarily disabled:
# VSN003
```

DUMP_VSNS_USED

(tape backups only) Specifies a file in which the VSNs of tapes that are used are written. By default, /dev/null is used, effectively disabling this feature.

DUMP_WORKSPACE

(disk and integrated backups only) Specifies a local directory used to store temporary files created during a dump process before moving them to the DUMP_DESTINATION filesystem. SGI recommends that you set this parameter for a disk backup that uses an NFS filesystem as its DUMP_DESTINATION filesystem. (The DUMP_WORKSPACE parameter is not normally required for integrated backups, but could be used if you require a staging area to tune performance.)

DUMP_XFSDUMP_PARAMS

(disk, tape, and integrated backups) Passes parameters to the xfsdump program. The value is not checked for validity, so you should use this parameter with care. Make sure that there are no conflicts with the xfsdump parameters generated by the DMF scripts. For integrated backups, the value set in a dumpgroup object or filesystem object overrides the value set here (the dumpgroup object has the highest precedence).

FILE_RETENTION_DAYS

(fast-mount cache only) Specifies the access age (in days) of a file that will be kept in the fast-mount cache during the process of freeing volumes. Files that have been accessed in fewer days will be moved onto another volume in the fast-mount cache.
The `run_fmc_free.sh` task will use the `dmemptytape`(8) command to move those files into another volume within the fast-mount cache before completely freeing the full volume. By default, all files within the volume are deleted.

Note: Be aware of the following when specifying this parameter:

- There may be a significant performance impact on the `run_fmc_free.sh` task.
- If the `dmmove`(8) command is used to move files and if there is another simultaneous `dmmove` active that is using the same *MOVE_FS* directory, the two processes will compete for the same disk space without any knowledge of each other. The result is that each process may encounter unexpected `ENOSPACE` errors. You can use the `FMC_MOVEFS` parameter to minimize this problem.

FMC_MOVEFS

(fast-mount cache only) Specifies the specific *MOVE_FS* scratch directory to be used by the `dmemptytape -f` option when moving files to another volume in the fast-mount cache, when required by the setting of `FILE_RETENTION_DAYS`. Also see `MOVE_FS` in "`dmdaemon` Object Parameters" on page 290.

FMC_NAME

(fast-mount cache only) Specifies the name of the `fastmountcache` object. See "`fastmountcache` Object Parameters" on page 390.

FREE_VOLUME_MINIMUM

(fast-mount cache only) Specifies the minimum percentage of free volumes in the fast-mount cache that must be available. When this threshold is reached, `run_fmc_free.sh` begins freeing full volumes in order to meet the percentage set for

FREE_VOLUME_TARGET. You should set FREE_VOLUME_MINIMUM so that it is less than the value for FREE_VOLUME_TARGET.

FREE_VOLUME_TARGET — *(fast-mount cache only)* Specifies the percentage of free volumes in the fast-mount cache that run_fmc_free.sh will try to achieve when the FREE_VOLUME_MINIMUM threshold is reached.

JOURNAL_RETENTION — Specifies the age of files that will be kept when the run_remove_journals.sh task removes journals. The run_remove_journals.sh task requires this parameter. This parameter is evaluated when run_remove_journals.sh is run. Valid values are an integer followed by one of:

m[inutes]
h[ours]
d[ays]
w[eeks]

You must specify a value that equates to at least seven days. (If you specify a value that equates to less time, DMF will round the value up to seven days.)

LOG_RETENTION — Specifies the age of files that will be kept when the run_remove_logs.sh task is run. This parameter is evaluated when run_remove_logs.sh is run. Valid values are an integer followed by one of:

m[inutes]
h[ours]
d[ays]
w[eeks]

You must specify a value that equates to at least seven days. (If you specify a value that equates to less time, DMF will round the value up to seven days.)

MAX_ALERTDB_SIZE — Specifies the maximum size of the alerts database. The run_remove_alerts.sh task uses but does not require this parameter. For more information, see the **Note** under ALERT_RETENTION and "Restrict the Size of the Alerts and Performance Records Databases" on

page 117. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

MAX_PERFDB_SIZE — Specifies the maximum size of the performance database. For more information, see the **Note** under PERF_RETENTION and "Restrict the Size of the Alerts and Performance Records Databases" on page 117. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

PERF_RETENTION — Specifies the age of performance records that are preferred to be kept when the run_remove_perf.sh task is run. This parameter is evaluated when run_remove_perf.sh is run. This task uses but does not require this parameter. Valid values are an integer followed by one of:

m[inutes]
h[ours]
d[ays]
w[eeks]

Note: The set of records chosen for deletion is the union of the records that are older than the PERF_RETENTION value and the oldest records when the database reaches the MAX_PERFDB_SIZE threshold size:

- If you specify PERF_RETENTION without MAX_ALERTDB_SIZE, older records will be deleted no matter how small the database is
- If you specify MAX_PERFDB_SIZE without PERF_RETENTION, records will be kept no matter how old they are, so long as the database remains below the threshold size
- If you specify neither parameter, no records will be deleted (and REMPERF_PARAMS is ignored)

PERF_RETENTION has no relationship to METRICS_RETENTION, the arena data, or PCP metrics.

REMALERT_PARAMS — Specifies additional parameters to be executed by run_remove_alerts.sh. The only possible value is

`-V`, which will remove unused space from the alerts database. By default, no space is removed.

Note: There is a performance penalty commensurate with the size of the reduction and other activity on the alerts database.

REMPERF_PARAMS

Specifies additional parameters to be executed by `run_remove_perf.sh`. The only possible value is `-V`, which will remove unused space from the performance database. By default, no space is removed.

Note: There is a performance penalty commensurate with the size of the reduction and other activity on the alerts database.

RUN_TASK

Specifies the tasks to be run. All of the RUN_TASK parameters have the same syntax in the configuration file:

`RUN_TASK $ADMINDIR/`*task_name time_expression*

The *task_name* is the script to be executed.

The *time_expression* defines when a task should be done. It is a schedule expression that the following form:

`[every` *n period*`] [at` *hh*`:`*mm*`[:`*ss*`] ...] [on` *day* `...]`

n is an integer.

period is one of:

```
minute[s]
hour[s]
day[s]
week[s]
month[s]
```

hh:mm:ss is hour, minutes, seconds.

day is a day of the month (1 through 31) or day of the week (sunday through saturday).

The following are examples of valid time expressions:

```
at 2:00
at 1:00 on tuesday
every 5 minutes
every day at 1:00
```

Note: When using an expression that contains both an every and an at clause, the valid values for *period* in the every clause are day[s] or week[s].

If you create your own scripts to be executed via the RUN_TASK parameter, be aware that DMF will equate $ADMINDIR to the appropriate directory, which is /usr/lib/dmf. When the task is run, DMF passes it the name of the object that requested the task and the name of the task group. The task itself may use the dmconfig(8) command to obtain further parameters from either of these objects.

You may comment-out the RUN_TASK parameters for any tasks you do not want to run.

SCAN_FILESYSTEMS — Specifies for the run_filesystem_scan.sh task the filesystems that dmscanfs(8) will scan. The default is to scan all managed filesystems.

SCAN_FOR_DMSTAT — Specifies for the run_filesystem_scan.sh task whether additional output files may be created (ON) or not (OFF). The default is ON.

If bit-file identifiers (BFIDs) and pathnames are included in the output file and SCAN_FOR_DMSTAT is enabled, an additional output file named bfid2path will be created in the daemon's *SPOOL_DIR* directory; this file is optimized for use by dmstat(8).

If file handles and BFIDs are in the output file and SCAN_FOR_DMSTAT is enabled, an additional output file named fhandle2bfid+path will be created in the

daemon's *SPOOL_DIR* directory; this file is optimized for use by dmemptytape(8).

SCAN_OUTPUT

Specifies for the run_filesystem_scan.sh task the name of the file into which dmscanfs will place output. The default is /tmp/dmscanfs.output.

This file, if it exists, is used by run_daily_report.sh and dmstat(8) and may be of use to site-written scripts or programs. Although DMF does not require this file, the output from run_daily_report.sh and dmstat will be incomplete if it is unavailable.

SCAN_PARALLEL

Specifies for the run_filesystem_scan.sh task whether dmscanfs will scan filesystems in parallel or not. You can specify one of the following:

ON	Enables parallel scanning on all available filesystems
number	Enables parallel scanning but limits the scan to the specified number of filesystems in order to reduce the load on system resources
OFF	Does not scan in parallel

Note: Enabling this parameter for a daemon task taskgroup may result in the filesystem scan completing in a shorter period of time, but it may also result in the task generating an unacceptable level of filesystem activity that interferes with user processes.

SCAN_PARAMS

Specifies additional dmscanfs parameters for the run_filesystem_scan.sh task. By default, dmscanfs is run with the -o stat option, which is suitable for use with run_daily_report.sh. SGI recommends that you use the default unless you require pathnames in the output or plan to use the dmstat(8) or dmemptytape(8) commands (which

require pathname for some operations); in these cases, SGI recommends that you set SCAN_PARAMS as follows:

```
SCAN_PARAMS        -o stat,path
```

If SCAN_PARAMS contains `-o all` or `-o path`, `dmscanfs` will do a recursive scan of the filesystems, which is much slower than the regular inode scan but results in pathnames being included in the output.

Note: SGI recommends that you do not specify the `-q` option (which suppresses the `dmscanfs` header line) as a value for SCAN_PARAMS because it makes the output file harder to parse with general-purpose scripts. The `run_daily_report.sh` task requires that this header line be present.

If BFIDs and pathnames are included in the output file and SCAN_FOR_DMSTAT is enabled, an additional output file named `bfid2path` will be created in the daemon's *SPOOL_DIR* directory. The `bfid2path` file is optimized for use by `dmstat`(8).

THRESHOLD — Specifies the integer percentage of active data on a volume. DMF will consider a volume to be sparse when it has less than this percentage of data that is still active.

TSREPORT_OPTIONS — Specifies for the `run_daily_tsreport.sh` task additional options that will be added to the end of the `tsreport` command line. For example, specifying `--host` will add an additional column with the hostname to the report. (This parameter is optional).

VOLUME_LIMIT	Specifies the maximum number of volumes that can be selected for merging at one time. By default, there is no limit.

taskgroup Object Examples

You can give the taskgroup object any name you like, but do not change the script names. You may comment-out the RUN_TASK parameters in the sample configuration files for any tasks you do not want to run. This section discusses the following:

- "taskgroup Object Example for Integrated Backups" on page 332
- "taskgroup Object Example for Tape Backups" on page 333
- "taskgroup Object Example for Disk Backups" on page 335
- "taskgroup Object Example for Disk Backups Using NFS"
- "taskgroup Object Example for Daemon Tasks" on page 336
- "taskgroup Object Example for Node Tasks" on page 339
- "taskgroup Object Example for Fast-Mount Cache Tasks" on page 339
- "taskgroup Object Example for Fast-Mount Cache Tasks Using File Retention" on page 340
- "taskgroup Object Example for Periodic dmmigrate Tasks" on page 341
- "taskgroup Object Example for Removing Alerts and Performance Records" on page 341

taskgroup Object Example for Integrated Backups

Example 7-9 taskgroup Object for Integrated Backups

```
define integrated_tasks
    TYPE                taskgroup
    RUN_TASK            $ADMINDIR/run_full_dump.sh on \
                                sunday at 00:01
    RUN_TASK            $ADMINDIR/run_partial_dump.sh on \
                                monday tuesday wednesday thursday \
                                friday saturday at 00:01
    RUN_TASK            $ADMINDIR/run_hard_deletes.sh \
```

```
                                at 23:00
    DUMP_DESTINATION    /dmf_backups
    DUMP_DR_MIRRORS     root@node1:/tmp/mymirrors
    DUMP_DR_EMAIL       admin@company.com

enddef
```

In the above example:

- The name of this task group is `integrated_tasks`. This can be any name you like, but it must be the same as one of the names provided for the `TASK_GROUPS` parameter of the `dmdaemon` object. See Example 7-4 on page 293.
- The `RUN_TASK` tasks specify the following:
  - The `run_full_dump.sh` task runs a full backup of managed filesystems each week on Sunday morning one minute after midnight.
  - The `run_partial_dump.sh` task backs up only those files in managed filesystems that have changed since the time a full backup was completed and is run each day of the week except Sunday, at one minute after midnight.
  - The `run_hard_deletes.sh` task removes soft-deleted entries from the DMF databases after the backup-media retention period, in this case 4 weeks. For more information, see "Cleaning Up Obsolete Database Entries" on page 586.
- The `/dmf_backups` filesystem must have a `filesystem` object, because it must be a managed filesystem that is already mounted.
- The DR bundle will be sent to the `/tmp/mymirrors` location on the machine named `node1` and to the `admin` email address

`taskgroup` Object Example for Tape Backups

Example 7-10 `taskgroup` Object for Tape Backups

```
define  dump_tasks
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_full_dump.sh on \
                                  sunday at 00:01
        RUN_TASK                $ADMINDIR/run_partial_dump.sh on \
                                  monday tuesday wednesday thursday \
                                  friday saturday at 00:01
        RUN_TASK                $ADMINDIR/run_hard_deletes.sh \
```

```
                                at 23:00
        DUMP_TAPES              HOME_DIR/tapes
        DUMP_RETENTION          4w
        DUMP_DEVICE             SILO_2
        DUMP_MIGRATE_FIRST      on
        DUMP_INVENTORY_COPY     /save/dump_inventory
enddef
```

In the above example:

- The name of this task group is `dump_tasks`. This can be any name you like, but it must be the same as the name provided for the `TASK_GROUPS` parameter of the `dmdaemon` object. See Example 7-4 on page 293.
- The `RUN_TASK` tasks specify the following:
  - The `run_full_dump.sh` task runs a full backup of managed filesystems each week on Sunday morning one minute after midnight.
  - The `run_partial_dump.sh` task backs up only those files in managed filesystems that have changed since the time a full backup was completed and is run each day of the week except Sunday, at one minute after midnight.
  - The `run_hard_deletes.sh` task removes soft-deleted entries from the DMF databases after the backup-media retention period, in this case 4 weeks. For more information, see "Cleaning Up Obsolete Database Entries" on page 586.
- The other parameters determine how the data from the filesystem backups will be managed:
  - `HOME_DIR` is defined in the `base` object (see "`base` Object" on page 277). For example, if `HOME_DIR` is `/dmf/home`, then `DUMP_TAPES` would resolve to `/dmf/home/dump_tasks/tapes`.
  - The DG that defines how to mount the tapes is `SILO_2`
  - The `dmmigrate` command will be run before the backups are taken
  - The XFS inventory files will be copied into `/save/dump_inventory`

taskgroup Object Example for Disk Backups

Example 7-11 taskgroup Object for Disk Backups

```
define  dump_tasks
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_full_dump.sh on \
                                  sunday at 00:01
        RUN_TASK                $ADMINDIR/run_partial_dump.sh on \
                                  monday tuesday wednesday thursday \
                                  friday saturday at 00:01
        RUN_TASK                $ADMINDIR/run_hard_deletes.sh \
                                  at 23:00
        DUMP_DESTINATION        /dmf/backups
        DUMP_MIRRORS            /mirror1 user@remotehost:/mirror2
        DUMP_RETENTION          4w
        DUMP_MIGRATE_FIRST      yes
        DUMP_COMPRESS           yes
enddef
```

The above example is similar to Example 7-10, page 333, except for the following:

- DUMP_DESTINATION rather than DUMP_TAPES specifies the location of the filesystem and database backup files. /dmf/backups must be a non-managed filesystem that is visible from the DMF server.
- Additional copies of the backup files will be placed in the following:
  - The /mirror1 directory, which is visible to the DMF server.
  - The remote mirror2 directory on the node named remotehost. The root user on the DMF server must be able to log in to remotehost as user using passwordless SSH.
- The backup files will be compressed using the default method (gzip -1).

taskgroup Object Example for Disk Backups Using NFS

Example 7-12 taskgroup Object for Disk Backups Using NFS

```
define  dump_tasks
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_full_dump.sh on \
                                  sunday at 00:01
```

```
        RUN_TASK                $ADMINDIR/run_partial_dump.sh on \
                                  monday tuesday wednesday thursday \
                                  friday saturday at 00:01
        RUN_TASK                $ADMINDIR/run_hard_deletes.sh \
                                  at 23:00
        DUMP_DESTINATION        /nfs/backups
        DUMP_WORKSPACE          /dumptmp
        DUMP_MIRRORS            /mirror1 user@remotehost:/mirror2
        DUMP_RETENTION          4w
        DUMP_MIGRATE_FIRST      yes
        DUMP_COMPRESS           yes
enddef
```

The above example is similar to Example 7-11, except that DUMP_DESTINATION is in an NFS filesystem, which requires that DUMP_WORKSPACE is set to a local filesystem in order to achieve best performance.

taskgroup Object Example for Daemon Tasks

Example 7-13 taskgroup Object for Daemon Tasks

```
define daemon_tasks
        TYPE                taskgroup
        RUN_TASK            $ADMINDIR/run_filesystem_scan.sh at 2:00
        RUN_TASK            $ADMINDIR/run_daily_report.sh at 3:00
        RUN_TASK            $ADMINDIR/run_daily_drive_report.sh at 4:00
        RUN_TASK            $ADMINDIR/run_audit.sh every day at 23:00
        RUN_TASK            $ADMINDIR/run_scan_logs.sh at 00:01
        RUN_TASK            $ADMINDIR/run_remove_logs.sh every \
                              day at 1:00
        RUN_TASK            $ADMINDIR/run_daily_tsreport.sh every \
                              day at 5:00
        LOG_RETENTION       4w
        RUN_TASK            $ADMINDIR/run_remove_journals.sh every \
                              day at 1:00
        JOURNAL_RETENTION   4w
        RUN_TASK            $ADMINDIR/run_copy_databases.sh \
                            every day at 3:00 12:00 21:00
        DATABASE_COPIES     /save/dmf_home /alt/dmf_home
enddef
```

In the above example:

- The name of this task group is `daemon_tasks`. This can be any name you like, but it must be the same as the name provided for the `TASK_GROUPS` parameter of the `dmdaemon` object. See Example 7-4 on page 293.
- The tasks specify the following:
  - At 2:00 AM, the `run_filesystem_scan.sh` task runs `dmscanfs`(8) on all managed filesystems and writes the output to `/tmp/dmscanfs.output` (using the defaults for `SCAN_FILESYSTEMS` and `SCAN_OUTPUT` because they are not specified).

    Because `SCAN_FOR_DMSTAT` (a misnomer) is not specified, its default value of `ON` means that the `fhandle2bfid+path` file will be created in the daemon's *SPOOL_DIR* directory because file handles and BFIDs are in the output file by default; however, the `bfid2path` file will not be created because by default pathnames are not included in the output file.
  - At 3:00 AM, the `run_daily_report.sh` task reports on DCM MSPs and managed filesystems (if `run_filesystem_scan.sh` has been run recently) and on all LSs.
  - At 4:00 AM, the `run_daily_drive_report.sh` task generates a report showing tape drives that have requested or required cleaning since the report was last run. If the time that the report was last run cannot be determined, or if this is the first time that the report was run, the reporting period is the previous 24 hours.

    The report uses information that the program `dmtscopy` copies from files in `/var/spool/ts/pd/log` to the directory *SPOOL_DIR*`/tspdlogs`. Only events from files in *SPOOL_DIR*`/tspdlogs` are reported. Information is not reported from tape drives that are not used with `ts`.
  - The `run_audit.sh` task runs `dmaudit` each day at 11:00 PM. If it detects any errors, the `run_audit.sh` task mails the errors to the e-mail address defined by the `ADMIN_EMAIL` parameter of the `base` object (described in "`base` Object" on page 277).
  - The `run_scan_logs.sh` task scans the DMF log files for errors at 12:01 AM. If the task finds any errors, it sends e-mail to the e-mail address defined by the `ADMIN_EMAIL` parameter of the `base` object.

– At 1:00 A.M., the `run_remove_logs.sh` task will remove logs that are more than 4 weeks old.

– At 5:00 AM, the `run_daily_tsreport.sh` task generates a report containing the output of the `tsreport` command. The reporting period covers the time since the task was last run. If that cannot be determined, the reporting period is the previous 24 hours.

  The report uses information that the program `dmtscopy` copies from files in `/var/spool/ts/pd/log` to the directory *SPOOL_DIR*`/tspdlogs`. Only events from files in *SPOOL_DIR*`/tspdlogs` are reported. Information is not reported from tape drives that are not used with `ts`.

  The task uses the following options for the `tsreport` command:

```
--noversion
--options
--wide
--tapestats
--drivestats
--errors
--tapealert
--startdate
--starttime
```

– At 1:00 A.M. the `run_remove_journals.sh` task removes journals that are more than 4 weeks old.

Note: The `run_remove_journals.sh` and `run_remove_logs.sh` tasks are not limited to the daemon journals and logs; they also clear the journals and logs for MSPs/LSs.

– The `run_copy_databases.sh` task makes a copy of the DMF databases each day at 3:00 AM, 12:00 noon, and 9:00 PM. The task copies a snapshot of the current DMF databases to either `/save/dmf_home` or `/alt/dmf_home`, whichever contains the oldest copy. Integrity checks are done on the databases before the copy is saved. If the checks fail, the copy is not saved, and the task sends e-mail to the address defined by the `ADMIN_EMAIL` parameter of the `base` object.

taskgroup Object Example for Node Tasks

Example 7-14 taskgroup Object for Node Tasks with Parallel DMF

```
define  node_tasks
    TYPE              taskgroup
    RUN_TASK          $ADMINDIR/run_remove_logs.sh every day at 1:00
    LOG_RETENTION     4w
enddef
```

In the above example:

- The name of this task group is node_tasks. This can be any name you like, but it must be the same as the name provided for the TASK_GROUPS parameter of the services object. See "services Object" on page 299.
- Log files more than 4 weeks old are deleted each day at 1:00 A.M.

  When using Parallel DMF, you should define the run_remove_logs.sh task for the taskgroup that applies to the node object rather than for the taskgroup that applies to the dmdaemon object.

  Note: The run_remove_logs.sh task is the only task available for service objects.

taskgroup Object Example for Fast-Mount Cache Tasks

Example 7-15 taskgroup Object for Fast-Mount Cache

```
define fmc_task
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_fmc_free.sh at 23:00
        FMC_NAME                copan_fmc
        FREE_VOLUME_MINIMUM     10
        FREE_VOLUME_TARGET      20
enddef
```

In the above example:

- The name of this task group is fmc_task. This can be any name you like, but it must be the same as the name provided for the TASK_GROUPS parameter of the dmdaemon object. See "dmdaemon Object" on page 289.

- The name of the fast-mount cache (`copan_fmc`) must match the name defined for the `fastmountcache` object. See "`fastmountcache` Object" on page 390.
- The `run_fmc_free.sh` task will be executed each day at 11:00 PM.
- When only 10% of the volumes in the fast-mount cache are free, DMF will free the volumes with the oldest write dates until 20% of the volumes are free.

`taskgroup` Object Example for Fast-Mount Cache Tasks Using File Retention

Example 7-16 `taskgroup` Object for Fast-Mount Cache Using File Retention

```
define fmc_task
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_fmc_free.sh at 23:00
        FMC_NAME                copan_fmc
        FREE_VOLUME_MINIMUM     10
        FREE_VOLUME_TARGET      20
        FILE_RETENTION_DAYS      3
        FMC_MOVEFS              /dmf/copanmove
enddef
```

In the above example:

- The `run_fmc_free.sh` task will be executed at the same time and using the same minimum threshold and target as in Example 7-15, page 339.
- Before deleting the data from a given volume, DMF will determine if any files should be retained in the fast-mount cache. If a volume to be freed contains files that have been accessed within the last 3 days, DMF will first move those files to another volume within the VG.

  Note: This may have a performance impact.

- If there are files that must be retained, the special scratch directory `/dmf/copanmove` will be used. This directory must be defined in the `MOVE_FS` parameter in the `dmdaemon` object. See "`dmdaemon` Object" on page 289.

taskgroup Object Example for Periodic dmmigrate Tasks

Example 7-17 taskgroup Object for Periodic dmmigrate Example

```
define dmmigrate_task
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_dmmigrate.sh every 4 hours
        DMMIGRATE_MINIMUM_AGE   25
        DMMIGRATE_WAIT          ON
enddef
```

In the above example:

- The name of the task is dmmigrate_task. If this task is called from the dmdaemon object, it will process all filesystems defined with automated space-management or user space-management enabled. If it is called from a specific filesystem or DCM msp object, it will migrate data from that object only.
- The run_dmmigrate.sh task will run every four hours.
- The minimum age of a file that will be migrated is 25 minutes.
- The task will wait for all migrations to complete before exiting.

taskgroup Object Example for Removing Alerts and Performance Records

Example 7-18 taskgroup Object for Removing Alerts and Performance Records

```
define remove_task
TYPE                    taskgroup

RUN_TASK                $ADMINDIR/run_remove_alerts.sh every day at 1:00
ALERT_RETENTION         3w
MAX_ALERTDB_SIZE        100m

RUN_TASK                $ADMINDIR/run_remove_perf.sh every day at 2:00
PERF_RETENTION          4w
MAX_PERFDB_SIZE         150m
REMPERF_PARAMS          -V

enddef
```

In the above example, two tasks will be run:

- `run_remove_alerts.sh` will be executed at 1:00 AM every day:
  - All alerts older than 3 weeks will be removed
  - If the alert database reaches 100 MB, the oldest records will be removed until the alert database is smaller than 100 MB
- `run_remove_perf.sh` will be executed at 2:00 AM every day:
  - All performance records older than 4 weeks will be removed
  - If the performance database reaches 150 MB, the oldest records will be removed until the performance database is smaller than 150 MB
  - The unused space in the performance database will be removed, despite the performance penalty

`dumpgroup` Object

This section discusses the following:

- "`dumpgroup` Object Name" on page 342
- "`dumpgroup` Object Summary" on page 343
- "`dumpgroup` Object Parameters" on page 343
- "`dumpgroup` Object Example for Integrated Backups" on page 345

See also "Precedence of Objects for Integrated Backups" on page 154.

`dumpgroup` Object Name

The name of the `dumpgroup` object is chosen by the administrator and may contain uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

dumpgroup Object Summary

Following is a summary of the potential parameters for the dumpgroup object:

```
dumpgroup
        TYPE
        DUMP_COMPRESS
        DUMP_FILE_SYSTEMS
        DUMP_MAX_FILESPACE
        DUMP_MIGRATE_FIRST
        DUMP_STREAMS
        DUMP_XFSDUMP_PARAMS
```

dumpgroup Object Parameters

Note: This stanza is used for only for integrated backups.

The setting of the dump parameters in the dumpgroup object overrides the value set by the filesystem object and taskgroup object for the specified filesystems. See "Precedence of Objects for Integrated Backups" on page 154.

Parameter	Description
TYPE	Specifies dumpgroup (required name for this type of object). There is no default.
DUMP_COMPRESS	*(integrated-backup context only)* Specifies the compression type and level to be used when dumping the filesystems specified by this object, overriding the value set by the filesystem and taskgroup objects. The following values are accepted:
	OFF — No compression.
	ON — Equivalent to gzip:1 (default).
	gzip[:*level*] — Use gzip(1) with the specified compression level. If you do not specify *level*, a value of 1 is used.

	`bzip2[:`*level*`]`	Use `bzip2`(1) with the specified compression level. If you do not specify *level*, a value of `9` is used.

For more information about legal values for *level*, see the man page for the compression tool.

For example, for a compression level of `3` with `bzip2`, you would use the following:

```
DUMP_COMPRESS       bzip2:3
```

Note: On backups consisting largely of migrated files, `gzip:1` (or `ON`) gives by far the best performance without sacrificing compression.

`DUMP_FILE_SYSTEMS` *(integrated-backup context only)* Specifies one or more filesystems to dump using the parameter values supplied by this object, overriding the values set by the `filesystem` and `taskgroup` objects. This parameter is required for a `dumpgroup` object.

`DUMP_MAX_FILESPACE` *(integrated-backup context only)* Specifies the maximum size of regular files to be included in a dump, overriding the value set for the `filesystem` and `taskgroup` objects. This limit does not apply to offline, dual-state, and partial-state files. Regular files using more than this space are silently left out of the backup. This value applies to all filesystems being dumped except for the DMF databases. The value is in bytes. You may also specify multiples of KB, MB, or GB by specifying a suffix of `k`, `m` or `g`, respectively; however, DMF will round up the value to 1 KB if you provide a value that equates to less than 1 KB. If this parameter is not provided, there is no limit.

This parameter can be dangerous because files are omitted silently and the limit applies to all of the filesystems specified by this object. Therefore SGI

recommends that you set this limit only for specific filesystems.

When this parameter is defined, you must set the `UPDATE_CTIME` parameter to `ON` in the `filesystem` object if you want to use incremental dumps for managed filesystems; see "`filesystem` Object Parameters" on page 351.

`DUMP_MIGRATE_FIRST` — *(integrated-backup context only)* Specifies whether or not the `dmmigrate` command is run before dumping the filesystems specified by this object, overriding the value set for the `filesystem` and `taskgroup` objects. Running `dmmigrate` first ensures that all migratable files in the managed filesystems are migrated, thus reducing the amount of backup media space and making it run much faster. You can set this parameter to `ON` or `OFF`. The default is `ON`.

`DUMP_STREAMS` — *(integrated-backup context only)* Specifies the number of `xfsdump` streams (threads) to use when dumping the filesystems specified by this object, overriding the value set for the `filesystem` and `taskgroup` objects. Using multiple streams can reduce backup and restore times. The default is `1` and the maximum is `20`.

`DUMP_XFSDUMP_PARAMS` — *(integrated-backup context only)* Passes parameters to the `xfsdump` program when dumping the filesystems specified by this object, overriding the value set for the `filesystem` and `taskgroup` objects. The value is not checked for validity, so you should use this parameter with care. Make sure that there are no conflicts with the `xfsdump` parameters generated by the DMF scripts.

`dumpgroup` Object Example for Integrated Backups

Example 7-19 `dumpgroup` Object Example for Integrated Backups

```
define  daemon
        TYPE                    dmdaemon
        MIGRATION_LEVEL         auto
        LS_NAMES                ls
        TASK_GROUPS             daemon_tasks dump_tasks
```

```
    ...
enddef

define  dump_tasks
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_full_dump.sh on sunday at 00:00
        RUN_TASK                $ADMINDIR/run_partial_dump.sh \
                                on monday tuesday wednesday thursday \
                                friday saturday at 00:00
        RUN_TASK                $ADMINDIR/run_hard_deletes.sh at 22:00
        DUMP_DESTINATION        /dmf/backups
        DUMP_RETENTION          4w
        DUMP_COMPRESS           on
        DUMP_GROUPS             dgr_admin dgr_cache
        DUMP_FILE_SYSTEMS       /dmfusr1 /dmfusr2
enddef

define  /dmfusr1
        TYPE                    filesystem
        MIGRATION_LEVEL         auto
        DUMP_COMPRESS           gzip:1
        POLICIES                space_policy vg_policy
enddef

define  dgr_admin
        TYPE                    dumpgroup
        DUMP_FILE_SYSTEMS       /dmf/home /dmf/openvault
        enddef

define  dgr_cache
        TYPE                    dumpgroup
        DUMP_FILE_SYSTEMS       /dmf/cache00
        DUMP_STREAMS            8
        DUMP_MAX_FILESPACE      1k
        UPDATE_CTIME            on
enddef
```

In the above example:

- The `taskgroup` named `dump_tasks` will run full and incremental backups on all of the managed filesystems
- The managed filesystems `/dmfusr1` and `/dmfusr2` as well as `/dmf/home`, `/dmf/openvault`, and `dmf/cache00` will be backed up because they are included in `DUMP_FILE_SYSTEMS` values in the `taskgroup` object and `dumpgroup` object

---

Note: The list of filesystems to be dumped is calculated as the union of the `DUMP_FILE_SYSTEMS` value in the `taskgroup` object and the `DUMP_FILE_SYSTEMS` value defined in each `dumpgroup` object that belongs to the `taskgroup`. If `DUMP_FILE_SYSTEMS` is defined in any object, only the explicitly-specified filesystems will be backed up; if `DUMP_FILE_SYSTEMS` is entirely absent from all objects, then all managed filesystems will be backed up by default.

- `dump_tasks` will use the default settings when dumping the managed filesystems, except for the following:
  - `/dmfusr`: the scripts will use a compression value of `gzip:1` and will dump files of any size
  - `/dmf/cache00`: regular files that exceed 1 KB will be excluded from the dump, and up to eight threads will be used when dumping the filesystem. This also requires that `UPDATE_CTIME` is enabled, to permit incremental backups of the managed filesystem.

`device` Object

This section discusses the following:

- "`device` Object Name" on page 348
- "`device` Object Summary" on page 348
- "`device` Object Parameters" on page 348

device Object Name

The name of the device object is chosen by the administrator and may contain uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

device Object Summary

Following is a summary of the potential parameters for the device object:

```
device
        TYPE
        MOUNT_SERVICE
        MOUNT_SERVICE_GROUP
        OV_ACCESS_MODES
        OV_INTERCHANGE_MODES
        TMF_TMMNT_OPTIONS
```

device Object Parameters

Normally, a drivegroup object defines the tape devices to be used by a taskgroup object (such as the example dump_tasks), with the LS and the backup scripts sharing the same devices. However, if backups are to use different drives from those in use by DMF, they should be defined by a device object. The parameters you define are based on the mounting service you intend to use.

Parameter	Description
TYPE	Specifies device (required name for this type of object). There is no default.
MOUNT_SERVICE	Specifies the mounting service. Possible values are openvault and tmf. You must use openvault for those DGs that contain tape drives on parallel data-mover nodes. The default is openvault.

Note: TMF is not supported on systems running the Red Hat Enterprise Linux operating system.

MOUNT_SERVICE_GROUP

Specifies the name by which the object's devices are known to the mounting service:

- OpenVault: use the OpenVault drive group name that is listed by the `ov_drivegroup` command. See "Use Corresponding Drive-Group Names in OpenVault and DMF" on page 121.
- TMF: use the device group name that would be used with the `-g` option on the `tmmnt` command. (TMF is not supported on systems running the Red Hat Enterprise Linux operating system.)

If this parameter is not specified, the `device` object's name is used.

OV_ACCESS_MODES

(OpenVault only) Specifies a list of access mode names that control how data is written to tape. For more information about the possible values, see the description of the `access` option in the `ov_mount`(8) man page.

Note: `xfsdump` does not use this parameter.

OV_INTERCHANGE_MODES

(OpenVault only) Specifies a list of interchange mode values that control how data is written to secondary storage. This optional parameter is applied when a volume is mounted or rewritten. By default, this list is empty.

Most drives support a value of either `compression` or `nocompression`.

For example, to specify that you want data compressed, use:

```
OV_INTERCHANGE_MODES      compression
```

Some drives support additional values. For example, the `T10000C` drive also supports the additional values `T10000C`, `T10000B`, and `T10000A`. For example, if you have a mixture of T10000C and T10000B drives, you could use the following to tell the T10000C drives to

write in compressed `T10000B` format so that both drives can then later read the same cartridges:

```
OV_INTERCHANGE_MODES      compression T10000B
```

For more information about the possible values, see the description of the `firstmount` option in the `ov_mount`(8) man page.

`TMF_TMMNT_OPTIONS` *(TMF only)* Specifies command options that should be added to the `tmmnt` command when mounting a tape. Because DMF uses the `-Z` option to `tmmnt`, options controlling block size and label parameters are ignored. Use `-g` if the group name is different from the `device` object's name. Use `-i` to request compression.

`filesystem` Object

This section discusses the following:

- "`filesystem` Object Name" on page 350
- "`filesystem` Object Summary" on page 351
- "`filesystem` Object Parameters" on page 351
- "`filesystem` Object Examples" on page 360

See also "Precedence of Objects for Integrated Backups" on page 154.

`filesystem` Object Name

The name of the `filesystem` object is the mount point. It may contain uppercase or lowercase alphanumeric characters, underscores, and the slash character (`/`). It cannot begin with an underscore or contain any white space.

Note: Do not use a symbolic link.

`filesystem` Object Summary

Following is a summary of the potential parameters for the `filesystem` object:

```
filesystem
        TYPE
        BUFFERED_IO_SIZE
        DIRECT_IO_SIZE
        DUMP_COMPRESS
        DUMP_MAX_FILESPACE
        DUMP_MIGRATE_FIRST
        DUMP_STREAMS
        DUMP_XFSDUMP_PARAMS
        IMMUTABLE_BFIDS
        MAX_MANAGED_REGIONS
        MESSAGE_LEVEL
        MIGRATION_LEVEL
        MIN_ARCHIVE_SIZE
        MIN_DIRECT_SIZE
        POLICIES
        POSIX_FADVISE_SIZE
        RELEASETHREADS
        TASK_GROUPS
        UPDATE_CTIME
        USE_AS_DUMP_DEST
        USE_UNIFIED_BUFFER
```

`filesystem` Object Parameters

You must have a `filesystem` object for each filesystem on which DMF can operate:

- A *managed filesystem* is a DMAPI-mounted XFS or CXFS filesystem on which DMF can migrate or recall files. (When using Parallel DMF, it must be a CXFS filesystem.) The object parameters specify the migration level for the filesystem, I/O options, and (if applicable) policies for MSP selection, file weighting, and automated space management.
- An *archive filesystem* is a POSIX filesystem (such as Lustre) that is not managed by DMF but from which you can efficiently copy files to secondary storage via the `dmarchive`(1) command. An archive filesystem does not support space management, migrations, or recalls. The `MIGRATION_LEVEL` parameter must be

set to `archive`. When using the Parallel DMF, such a filesystem should be mounted on the DMF server and all mover nodes.

Note: For integrated backups, the value set in the `dumpgroup` object overrides the value set here. See:

- "Precedence of Objects for Integrated Backups" on page 154
- "`dumpgroup` Object Parameters" on page 343

The `filesystem` object parameters are as follows:

Parameter	Description
`TYPE`	Specifies `filesystem` (required name for this type of object). There is no default.
`BUFFERED_IO_SIZE`	Specifies the size of I/O requests when reading from or writing to this filesystem using buffered I/O. The legal range of values is `4096`–`16777216`. The default is `262144`. By default, the unit of measure is bytes; see "Units of Measure" on page 275. However, this parameter is ignored when recalling files if `USE_UNIFIED_BUFFER` is set to `ON` (which is the default).
`DIRECT_IO_SIZE`	Specifies the size of I/O requests when reading from this filesystem using direct I/O. The legal range of values is `65536`–`18446744073709551615`. The default value depends on the filesystem's configuration, but will not exceed the value of `DIRECT_IO_MAXIMUM_SIZE` defined in the `base` object (see "`base` Object" on page 277). By default, the unit of measure is bytes; see "Units of Measure" on page 275. This parameter is ignored if the filesystem does not support direct I/O. For more information about direct I/O, see `O_DIRECT` in the `open`(2) man page.
`DUMP_COMPRESS`	*(integrated-backup context only)* Specifies the compression type and level to be used when dumping this filesystem, overriding the value set for the `taskgroup` object. The following values are accepted: `OFF` No compression.

`ON`	Equivalent to `gzip:1` (default).
`gzip[:`*level*`]`	Use `gzip`(1) with the specified compression level. If you do not specify *level*, a value of `1` is used.
`bzip2[:`*level*`]`	Use `bzip2`(1) with the specified compression level. If you do not specify *level*, a value of `9` is used.

For more information about legal values for *level*, see the man page for the compression tool.

For example, for a compression level of `3` with `bzip2`, you would use the following:

```
DUMP_COMPRESS       bzip2:3
```

Note: On backups consisting largely of migrated files, `gzip:1` (or `ON`) gives by far the best performance without sacrificing compression.

For integrated backups, the value set in a `dumpgroup` object overrides the value set here.

`DUMP_MAX_FILESPACE` *(integrated-backup context only)* Specifies the maximum size of regular files to be included when dumping this filesystem, overriding the value set for the `taskgroup` object. (This limit does not apply to offline, dual-state, and partial-state files.) Regular files using more than this space are silently left out of the backup. The value is in bytes. You may also specify multiples of KB, MB, or GB by specifying a suffix of `k`, `m`. or `g`, respectively; however, DMF will round up the value to 1 KB if you provide a value that equates to less than 1 KB. If this parameter is not provided, there is no limit.

When this parameter is defined, you must set the UPDATE_CTIME parameter to ON in the filesystem object if you want to use incremental dumps; see "filesystem Object Parameters" on page 351.

For integrated backups, the value set in a dumpgroup object overrides the value set here. This parameter can be dangerous because files are omitted silently.

DUMP_MIGRATE_FIRST — *(integrated-backup context only)* Specifies whether or not the dmmigrate command is run before dumping this filesystem, overriding the value set for the taskgroup object. Running dmmigrate first ensures that all migratable files in the managed filesystem are migrated, thus reducing the amount of backup media space and making it run much faster. You can set this parameter to ON or OFF. The default is ON. For integrated backups, the value set in a dumpgroup object overrides the value set here.

DUMP_STREAMS — *(integrated-backup context only)* Specifies the number of xfsdump streams (threads) to use when dumping this filesystem, overriding the value set for the taskgroup object. Using multiple streams can reduce backup and restore times. The default is 1 and the maximum is 20. For integrated backups, the value set in a dumpgroup object overrides the value set here.

DUMP_XFSDUMP_PARAMS — *(integrated-backup context only)* Passes parameters to the xfsdump program when dumping this filesystem, overriding the value set for the taskgroup object. The value is not checked for validity, so you should use this parameter with care. Make sure that there are no conflicts with the xfsdump parameters generated by the DMF scripts. For integrated backups, the value set in a dumpgroup object overrides the value set here.

IMMUTABLE_BFIDS — Specifies whether or not DMF should restrict operations that change the BFIDs of the migrated files in this filesystem. When enabled, the DMF daemon will reject dmunput and dmmvtree requests related to this filesystem. (It is possible for BFIDs to change, however, if you use the dmaudit command to correct cases of

multiple files with the same BFID.) You can set this parameter to `ON` or `OFF`. The default is `OFF` (disables). You must set this parameter to `ON` if you are using a product that stores BFIDs externally to DMF, such as Arcitecta Mediaflux; in that case, see the product documentation for how to use this parameter and `dmaudit`.

`MAX_MANAGED_REGIONS` Sets the maximum number of managed regions that DMF will assign to a file in this filesystem. You can set `MAX_MANAGED_REGIONS` to any number that is less than the actual number of regions that will fit in the filesystem attribute. For a XFS or CXFS filesystem, that number is 3275.

By default, DMF allows a DMF attribute to contain the maximum number of managed regions that will still allow the attribute to fit completely inside the inode, based on inode size and `attr` type. The default value for a `filesystem` object that does not have a `MAX_MANAGED_REGIONS` parameter is calculated at filesystem mount time. This value is chosen to ensure that the DMF attribute will fit inside the inode, assuming that no other attribute (such as an ACL) is already occupying the inode's attribute space. Table 5-1 on page 172, lists the default maximum file regions. This parameter does not apply to a filesystem with a `MIGRATION_LEVEL` of `archive`.

Caution: You should use `MAX_MANAGED_REGIONS` cautiously. If you set this parameter to a value that is larger than the default maximum (see Table 5-1 on page 172), the DMF attribute may not fit inside the inode. If there are many files with DMF attributes outside of the inode, filesystem scan times can increase greatly. To avoid this problem, SGI recommends that a file that has exceeded the maximum default file regions be made offline (that is, having a single region) as soon as possible after the online data has been accessed.

MESSAGE_LEVEL

Specifies the highest message level that will be written to the automated space-management log (autolog). It must be an integer in the range 0-6; the higher the number, the more messages written to the log file. The default is 2. This parameter applies only to a filesystem with a MIGRATION_LEVEL of auto. For more information on message levels, see Chapter 10, "Message Log Files" on page 507.

MIGRATION_LEVEL

Specifies the level of migration services for this filesystem. (Recall from offline media is not affected by the value of MIGRATION_LEVEL.) Valid values are:

- archive (only for DMF direct archiving via the dmarchive command)
- auto (automated space management)
- none (no migration)
- user (only user-initiated migration using the dmput or dmmigrate commands)

The migration level actually used for this filesystem is the lesser of the MIGRATION_LEVEL value for the dmdaemon object and this value. If you do not want automated space-management for a filesystem, set MIGRATION_LEVEL to user or none. The default is auto.

When using Parallel DMF, a managed filesystem (that is, a filesystem where DMF can migrate or recall files) must be a CXFS filesystem.

MIN_ARCHIVE_SIZE

Specifies the minimum file size required for the dmarchive command to copy data directly between an archive filesystem and DMF secondary storage. Files smaller than this size will instead be copied between the archive filesystem and a managed filesystem before being possibly migrated or recalled from DMF secondary storage. The legal range of values is 1–18446744073709551615. The default is 1 (meaning that all files will be copied directly to secondary storage with only metadata going to the

managed filesystem). By default, the unit of measure is bytes; see "Units of Measure" on page 275. This parameter applies only to a filesystem with a `MIGRATION_LEVEL` value of `archive`.

`MIN_DIRECT_SIZE` — Determines whether direct or buffered I/O is used when reading from this filesystem. If the number of bytes to be read is smaller than the value specified, buffered I/O is used; otherwise, direct I/O is used. The legal range of values is `0` (direct I/O is always used) through `18446744073709551615` (direct I/O is never used). The default is `0`. By default, the unit of measure is bytes; see "Units of Measure" on page 275. This parameter is ignored if the filesystem does not support direct I/O or is a real-time filesystem. For more information about direct I/O, see `O_DIRECT` in the `open`(2) man page.

Note: Buffered I/O is always used when writing to a filesystem.

`POLICIES` — Specifies the names of the configuration objects defining policies for this filesystem. Policies are defined with `policy` objects (see "`policy` Object"). The `POLICIES` parameter is required; there is no default value. A policy can be unique to each managed filesystem or it can be reused numerous times. This parameter does not apply to a filesystem with a `MIGRATION_LEVEL` of `archive`.

`POSIX_FADVISE_SIZE` — Specifies the number of bytes after which DMF will call `posix_fadvise()` with advice `POSIX_FADV_DONTNEED` when recalling files. The minimum is `0`, which means that `posix_fadvise` is never used. The maximum is `18446744073709551615`. The default is `100000000`, which will call `posix_fadvise` after each 100,000,000 bytes (approximately) it has written to the file. By default, the unit of measure is bytes; see "Units of Measure" on page 275. If you notice performance problems when recalling files, changing this value may

make an improvement, but this will require testing; see "Poor Recall Performance" on page 669.

DMF does not synchronize the file at this point. If POSIX_FADVISE_SIZE is set to a nonzero value, DMF will also call posix_fadvise when a region is made online.

Note: Setting this parameter to a small, nonzero value may have an adverse affect on performance. See the posix_fadvise(2) man page for more information.

RELEASETHREADS

Specifies the number of threads in the DMF daemon that will be created to handle requests to release filesystem blocks when files are made offline (OFL). The default and minimum value is 1. The maximum value is 20. This parameter does not apply to filesystems with a MIGRATION_LEVEL of archive.

TASK_GROUPS

Names the taskgroup objects that contain tasks the daemon should run when MIGRATION_LEVEL is set to auto. By default, no tasks are run.

UPDATE_CTIME

Specifies whether or not the st_ctime value of files in this filesystem should be changed when certain operations are performed. (For more information about st_ctime, see the stat(2) man page.) If set to ON, DMF will update the st_ctime value of a file under the following circumstances:

- When the file transitions from a migrating state (MIG or ARC) to a migrated state (DUL, PAR, or OFL)
- When the file transitions from a migrated state or a migrating state to regular state (REG)
- When the file's sitetag value changes (for more information, see the dmtag(1) man page)

You should set this parameter to ON if you have defined DUMP_MAX_FILESPACE and wish to perform incremental backups. See:

- "dumpgroup Object Parameters" on page 343
- "filesystem Object Parameters" on page 351
- "taskgroup Object Parameters" on page 312
- "DUMP_MAX_FILESPACE Requires UPDATE_CTIME for Incremental Backups" on page 148

This parameter is valid only on managed filesystems. You can set this parameter to ON or OFF. The default is OFF.

USE_AS_DUMP_DEST — *(integrated backups only)* Determines if this filesystem can be used for storing dumps. The value can be one of the following:

- ON, which means that DMF can be configured in a taskgroup object to use this filesystem for storing dumps. You can set this parameter to ON for only one filesystem object in the DMF configuration. This filesystem should not be used for holding ordinary user data and SGI recommends that the MIGRATION_LEVEL value for this filesystem is set to auto.
- OFF, which means that DMF will not use this filesystem for storing dumps and therefore you cannot configure this filesystem as a DUMP_DESTINATION value in any dump taskgroup. The default setting is OFF.

USE_UNIFIED_BUFFER — Determines how DMF manages its buffers when recalling files in this filesystem. The value can be one of the following:

- ON, which means that DMF will use the same buffer for reading and writing and BUFFERED_IO_SIZE is ignored when recalling files. Setting the value to ON will cause the size of I/O requests to be small when recalling data from a cloud, DCM, disk, or FTP MSP. The default setting is ON.
- OFF, which means that DMF uses separate buffers for reading and writing during recall. That is, DMF reads data from its backing store (such as tape) into a buffer and then copies the data into another buffer for writing. An additional thread for writing is also used.

`filesystem` Object Examples

This section discusses the following examples:

- "`filesystem` Object Example for a Managed Filesystem" on page 361
- "`filesystem` Object Example for Integrated Backups" on page 362
- "`filesystem` Object Example for Incremental Backups that Restrict File Size" on page 362
- "`filesystem` Object for DMF Direct Archiving" on page 363

`filesystem` Object Example for a Managed Filesystem

Example 7-20 defines a `filesystem` object named `/user1` using default values except as noted.

Example 7-20 `filesystem` Object for a Managed Filesystem

```
define  /user1
        TYPE                    filesystem
        MIGRATION_LEVEL         user
        POLICIES                fs_msp
enddef
```

In the above example:

- The `define` parameter must have a value that is the mount point of the filesystem you want DMF to manage, in this case `/user1`. Do not use a symbolic link.
- Only user-initiated migration will be used for migration to offline media.
- The migration policy is set by the `policy` object named `fs_msp`. See "`policy` Object" on page 364.
- The example uses the default value for the following parameters, which also apply to a `filesystem` object for a managed filesystem:

  ```
  BUFFERED_IO_SIZE
  DIRECT_IO_SIZE
  MAX_MANAGED_REGIONS
  MESSAGE_LEVEL
  MIN_DIRECT_SIZE
  POSIX_FADVISE_SIZE
  TASK_GROUPS
  USE_UNIFIED_BUFFER
  ```

`filesystem` Object Example for Integrated Backups

Example 7-21 defines a `filesystem` object that will be used for integrated backups.

Example 7-21 `filesystem` Object for Integrated Backups

```
define  /dmf/backups
    TYPE                filesystem
    MIGRATION_LEVEL     auto
    POLICIES            backup_space_policy backup_vg_policy
    USE_AS_DUMP_DEST    on
enddef
```

In the above example:

- The `define` parameter must have a value that is the mount point of the filesystem you want to use for backups, in this case `/dmf/backups`. Do not use a symbolic link.
- Automatic migration will be used.
- The migration policy is set by the `policy` objects named `backup_space_policy` and `backup_vg_policy`. `backup_vg_policy` should include at least two VGs to which data from this filesystem will be migrated . See "`policy` Object" on page 364.
- This filesystem can be configured as a `DUMP_DESTINATION` for an integrated-backup `taskgroup` object. DMF will then use this filesystem for storing dumps.

`filesystem` Object Example for Incremental Backups that Restrict File Size

Example 7-22 defines three `filesystem` objects in a configuration that uses incremental backups. As an illustration, the objects for `/user1` and `/user2` restrict the size of regular files included in the dumps in order to make the backup completion times more consistent.

Example 7-22 `filesystem` Objects for Incremental Backups that Restrict File Size

```
define  /user1
        TYPE                    filesystem
        MIGRATION_LEVEL         user
        POLICIES                backup_space_policy backup_vg_policy
```

```
        DUMP_MAX_FILESPACE      1k
        UPDATE_CTIME            on
enddef

define  /user2
        TYPE                    filesystem
        MIGRATION_LEVEL         user
        POLICIES                backup_space_policy backup_vg_policy
        DUMP_MAX_FILESPACE      1k
        UPDATE_CTIME            on
        enddef

define  /user3
        TYPE                    filesystem
        MIGRATION_LEVEL         user
        POLICIES                backup_space_policy backup_vg_policy
        enddef
```

The above example is similar to previous examples, except for the following:

- Regular files that exceed 1 KB will be excluded from the dumps for `/user1` and `/user2` (no regular files will be excluded for `/user3`).
- The `st_ctime` value will be changed on `/user1` and `/user2` when certain operations are performed, as described in "`filesystem` Object Parameters" on page 351. This allows the `run_partial_dump.sh` script to perform incremental backups and ensures that the regular files in those filesystems will not be missed. (`UPDATE_CTIME` does not need to be set for `/user3` because its object does not have `DUMP_MAX_FILESPACE` set.)

Note: If `UPDATE_CTIME` was **not set** for the `/user1` and `/user2` objects, then `run_partial_dump.sh` would always perform full (level-0) backups for those managed filesystems.

`filesystem` Object for DMF Direct Archiving

Example 7-23 defines a `filesystem` object for an archive filesystem named `/lustrefs`, using default values except as noted:

Example 7-23 `filesystem` Object for DMF Direct Archiving

```
define  /lustrefs
        TYPE                    filesystem
        MIGRATION_LEVEL         archive
        MIN_ARCHIVE_SIZE        262144
enddef
```

In the above example:

- The `define` parameter must have a value that is the mount point of the archive filesystem, in this case `/lustrefs`. Do not use a symbolic link.
- File data in `/lustrefs` can be copied directly to secondary storage by users via the `dmarchive`(1) command.
- Files that are smaller than 262,144 bytes are never archived via `dmarchive` but instead will be copied to a managed filesystem before being possibly migrated or recalled from DMF secondary storage.
- The example uses the default value for the following parameters, which also apply to a `filesystem` object for DMF direct archiving:

  ```
  BUFFERED_IO_SIZE
  DIRECT_IO_SIZE
  MIN_DIRECT_SIZE
  POSIX_FADVISE_SIZE
  USE_UNIFIED_BUFFER
  ```

`policy` Object

This section discusses the following:

- "Functions of `policy` Parameters" on page 365
- "Rules for `policy` Parameters" on page 366
- "`policy` Object Name" on page 368
- "Managed Filesystem `policy` Parameters" on page 368
- "DCM MSP `STORE_DIRECTORY` `policy` Parameters" on page 375
- "`when` Clause" on page 380

- "`ranges` Clause" on page 383
- "`policy` Configuration Examples" on page 385

Functions of `policy` Parameters

A `policy` object specifies behavior for managing the following:

- A managed filesystem
- A DCM `STORE_DIRECTORY`

The `policy` object parameters specify the following functions:

- "Automated Space-Management Overview" on page 365
- "File Weighting Overview" on page 365
- "MSP/VG Selection Overview" on page 366

For details about the parameters, see:

- "Managed Filesystem `policy` Parameters" on page 368
- "DCM MSP `STORE_DIRECTORY` `policy` Parameters" on page 375

Automated Space-Management Overview

DMF lets you automatically monitor filesystems and migrate data as needed to prevent filesystems from filling. This capability is implemented by the `dmfsmon`(8) daemon. After the `dmfsmon` daemon has been initiated, it will begin to monitor the managed filesystem in order to maintain the level of free space specified in the configuration file.

Note: Ideal values for these parameters are highly site-specific, based largely on filesystem sizes and typical file sizes.

File Weighting Overview

When DMF is conducting automated space management, it derives an ordered list of files (called a *candidate list*) and migrates or frees files starting at the top of the list. The ordering of the candidate list is determined by weighting factors that are defined by parameters in the configuration file. You can use the file weighting parameters

multiple times to specify that different files should have different weights. For more details, see Chapter 11, "Automated Space Management" on page 509.

MSP/VG Selection Overview

DMF can be configured to have many MSPs/VGs, including those specified in an MG (see "`migrategroup` Object" on page 426).

Each MSP/VG manages its own set of volumes. The MSP/VG selection parameters let you migrate files with different characteristics to different MSPs/VGs. You can use the MSP/VG selection parameters multiple times to specify that different files should have different MSP/VG selection values.

Rules for `policy` Parameters

This section discusses the following:

- "Managed Filesystem Rules" on page 366
- "DCM MSP `STORE_DIRECTORY` Rules" on page 367

Managed Filesystem Rules

The rules for a `policy` object that is migrating a managed filesystem are as follows:

- The `POLICIES` parameter for a `filesystem` object must specify one and only one MSP/VG selection policy.
- The `TYPE` parameter is required for any `policy` object:

Parameter	Description
`TYPE`	Specifies `policy` (required name for this type of object). There is no default.

- If the `MIGRATION_LEVEL` parameter for a `filesystem` object is `auto`, the `POLICIES` parameter for that object must specify one and only one space-management policy.
- You do not need to specify a weighting policy if the default values are acceptable.

- Providing the above rules are followed, you can have many different combinations of policies. For example, you could configure one policy that defines all three categories of policy parameters (automated space management, MSP/VG selection, and file weighting) and share that policy among all the filesystems, or you could configure any number of individual MSP/VG selection policies and space-management policies (including weighting parameters) that you can apply to one or more filesystems.

DCM MSP STORE_DIRECTORY Rules

The rules for a `policy` object that is managing a DCM MSP `STORE_DIRECTORY` are as follows:

- The `TYPE` parameter is required for any `policy` object:

Parameter	Description
`TYPE`	Specifies `policy` (required name for this type of object). There is no default.

- If the `MIGRATION_LEVEL` for a `filesystem` object is `auto`, the `POLICIES` parameter for that object must specify one and only one space-management policy.
- You do not need to specify a weighting policy if the default values are acceptable.
- You can configure one policy that defines all three categories of policy parameters (automated space management, file weighting, and VG selection) and share that policy among all the filesystems. Alternatively, you might create a VG selection policy for all filesystems and a space-management policy (including weighting parameters) for all filesystems.
- The DCM MSP supports the concept of *dual-residence*, which means that a cache-resident copy of a migrated file has already been copied to secondary storage and can therefore be released quickly in order to prevent the cache filling, without any need to first copy it to secondary storage. It is analogous to a dual-state file that is managed by the standard disk MSP and has equivalent policy parameters to control it.
- The age and space weighting parameters refer to the copies in the cache, not the originals in the managed filesystem.

`policy` Object Name

The name of the `policy` object is chosen by the administrator and may contain uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

Managed Filesystem `policy` Parameters

This section discusses the following:

- "`policy` Object Summary for a Managed Filesystem" on page 368
- "Automated Space-Management Parameters for a Managed Filesystem" on page 369
- "File Weighting Parameters for a Managed Filesystem" on page 371
- "MSP/VG Selection Parameters for a Managed Filesystem" on page 374

`policy` Object Summary for a Managed Filesystem

Following is a summary of the potential parameters for the `policy` object for a managed filesystem:

```
policy
        TYPE
        AGE_WEIGHT
        FREE_DUALSTATE_FIRST
        FREE_SPACE_DECREMENT
        FREE_SPACE_MINIMUM
        FREE_SPACE_TARGET
        MIGRATION_TARGET
        SELECT_MSP  or  SELECT_VG
        SITE_SCRIPT
        SPACE_WEIGHT
```

For details, see:

- "Automated Space-Management Parameters for a Managed Filesystem" on page 369
- "File Weighting Parameters for a Managed Filesystem" on page 371
- "MSP/VG Selection Parameters for a Managed Filesystem" on page 374

Automated Space-Management Parameters for a Managed Filesystem

The following parameters control automated space management for a managed filesystem:

Parameter	Description
FREE_DUALSTATE_FIRST	When set to ON, specifies that dmfsfree will first free dual-state and partial-state files before freeing files it must migrate. The default is OFF.
FREE_SPACE_DECREMENT	Specifies the integer percentage of filesystem space by which dmfsmon or dmdskmsp will decrement FREE_SPACE_MINIMUM if it cannot find enough files to migrate so that the value is reached. The decrement is applied until a value is found that can be achieved. If space later frees up, the FREE_SPACE_MINIMUM is reset to its original value. Valid values are in the range 1 through the value of FREE_SPACE_TARGET. The default is 2.
FREE_SPACE_MINIMUM	Specifies the minimum integer percentage of the total filesystem space that dmfsmon tries to maintain as free. When the available free space reaches or falls below this threshold value, dmfsmon will begin to migrate files (freeing data for dual-state files as needed) in order to meet the percentages set for FREE_SPACE_MINIMUM, FREE_SPACE_TARGET, and MIGRATION_TARGET. One and only one of the policy stanzas associated with a given filesystem stanza should have a FREE_SPACE_MINIMUM value if the stanza has a MIGRATION_LEVEL of auto; in this case, there is no default value. You should set FREE_SPACE_MINIMUM so that it is less than the values for FREE_SPACE_TARGET and MIGRATION_TARGET. Figure 7-1 describes the concepts of free space and migration targets, using as an example a minimum free-space threshold of 10%. For example, if offline files are recalled or regular files are added to the filesystem such that only 10% of it is free, DMF will try to reach the free-space target of 30% by freeing the space

currently held by dual-state files and try to reach the migration target of 80% by migrating regular files so that they become dual-state.

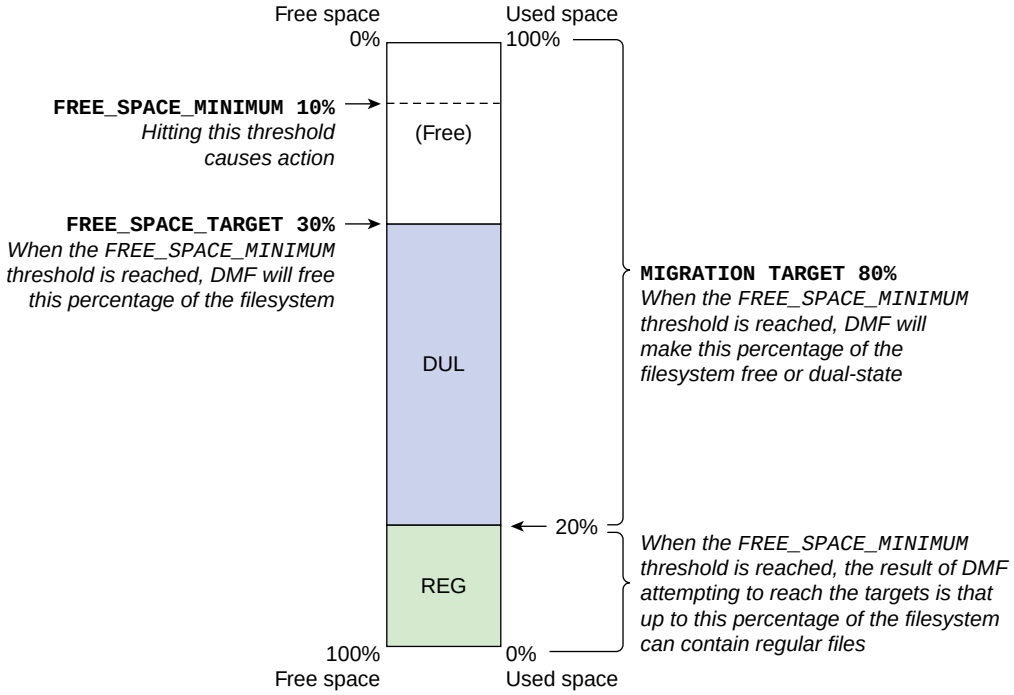

Figure 7-1 Concepts of Free-Space Minimum and Target

For more details, see Chapter 11, "Automated Space Management" on page 509.

For the information on how this parameter is used when automated space management is not configured, see the `dmf.conf`(5) man page.

`FREE_SPACE_TARGET` Specifies the integer percentage of total filesystem space that `dmfsfree` or `dmdskfree` tries to free if free space reaches or falls below the `FREE_SPACE_MINIMUM` threshold. You should set `FREE_SPACE_TARGET` so

	that it is less than MIGRATION_TARGET. One and only one of the policy stanzas associated with a given filesystem stanza should have a FREE_SPACE_TARGET value if the stanza has a MIGRATION_LEVEL of auto; in this case, there is no default value.
MIGRATION_TARGET	Specifies the integer percentage of total filesystem space that dmfsmon tries to maintain as a reserve of space that is free or occupied by dual-state files. (The online space occupied by dual-state files can be freed quickly if free space reaches or falls below FREE_SPACE_MINIMUM.) One and only one of the policy stanzas associated with a given filesystem stanza should have a MIGRATION_TARGET value if the stanza has a MIGRATION_LEVEL of auto; in this case, there is no default value.
SITE_SCRIPT	Specifies the site-specific script to execute when the dmfsfree, dmdskfree, or dmfsmon command is run: • If the script returns a zero exit status, the command continues its normal processing • If the script returns a nonzero exit status, the command returns immediately, using this value as its own exit status See dmfsfree(8) or dmdskfree(8) for further details. This parameter is optional.

For more details, see Chapter 11, "Automated Space Management" on page 509.

See also:

- "Functions of policy Parameters" on page 365
- "Managed Filesystem Rules" on page 366

File Weighting Parameters for a Managed Filesystem

The following parameters control file weighting for a managed filesystem:

Parameter	Description
AGE_WEIGHT	Specifies a floating-point constant and floating-point multiplier to use when calculating the weight given to a file's age, calculated as follows: *constant* + (*multiplier* * *file_age_in_days*) The default is a *constant* of 1 and a *multiplier* of 1. The AGE_WEIGHT parameter accepts an optional when clause, which contains a conditional expression. You can use this clause to select which files should use the AGE_WEIGHT values. See "when Clause" on page 380. The AGE_WEIGHT parameter also accepts an optional ranges clause, which specifies the ranges of a file for which the parameter applies. See "ranges Clause" on page 383. DMF checks each AGE_WEIGHT parameter in turn, in the order that they occur in the configuration file. If the when clause is present and no ranges clause is present, DMF determines whether the file matches the criteria in the clause. If no when clause is present, a match is assumed. If the file matches the criteria, the file weight is calculated from the parameter values. If they do not match, the next instance of that parameter is examined. You can configure a negative value to ensure that specific files are never automatically migrated. For example, you might want to set a minimum age for migration. The following parameter specifies that files that have been accessed within 1 day are never automatically migrated: `AGE_WEIGHT -1  0.0  when age <= 1` **Note:** DMF calculates the age weight and space weight separately. If either value is less than zero, the file is **not** automatically migrated and the file or range is **not** automatically freed. Otherwise, the two values are summed to form the file's or range's weight.

`SPACE_WEIGHT`

Specifies a floating-point constant and floating-point multiplier to use when calculating the weight given to a file's size, calculated as follows:

constant `+` (*multiplier* `*` *file_disk_space_in_bytes*)

The default is a *constant* of `0` and a *multiplier* of `0`.

For a partial-state file, *file_disk_space_in_bytes* is the amount of space occupied by the file at the time of evaluation.

The `SPACE_WEIGHT` parameter accepts an optional `when` clause, which contains a conditional expression. See "`when` Clause" on page 380.

The `SPACE_WEIGHT` parameter also accepts an optional `ranges` clause, which specifies the ranges of a file for which the parameter applies. See "`ranges` Clause" on page 383.

You can configure a negative value to ensure that specific files are never automatically migrated. For example, you might want to set a minimum size for migration. The following parameter specifies that small files are never automatically migrated:

```
SPACE_WEIGHT -1    0    when space <= 4k
```

Note: DMF calculates the age weight and space weight separately. If either value is less than zero, the file is **not** automatically migrated and the file or range is **not** automatically freed. Otherwise, the two values are summed to form the file's or range's weight.

See also:

- "Functions of `policy` Parameters" on page 365
- "Managed Filesystem Rules" on page 366

MSP/VG Selection Parameters for a Managed Filesystem

The following parameters control MSP/VG selection for a managed filesystem:

Parameter	Description
SELECT_MSP, SELECT_VG	Specifies the MSPs, VGs, and MGs to use for migrating a file. If you use an MG, you must not specify overlapping MSPs, VGs, or MGs on the same SELECT_VG and SELECT_MSP statement (taking care to ensure that the statement expands to a set of non-overlapping MSPs and VGs when all of the group members of the MGs are considered). See "migrategroup Object" on page 426. **Note:** The parameters are not used for defining which MSP/VG to use for recalls; for that, see the definitions of the LS_NAMES, MSP_NAMES, DRIVE_GROUPS, and VOLUME_GROUPS parameters. The SELECT_MSP and SELECT_VG parameters are equivalent. VGs, cloud MSPs, disk MSPs, FTP MSPs, and MGs may be specified by either parameter. You can list as many MSP/VG/MG names as you have msp, volumegroup, and migrategroup objects defined. A copy of the file will be migrated as follows: • To each MSP/VG listed explicitly • For each MG listed, to exactly one MSP/VG that is a member of the MG

Note: If DUMP_WORKSPACE is not set or if it is explicitly set to a location inside the backup filesystem (the filesystem that has USE_AS_DUMP_DEST set to ON), then you must prevent the migration of temporary files produced by the integrated-backup method. These temporary files have a project ID of 4136. To accomplish this, set the **first** SELECT_VG/SELECT_MSP parameter to the following value:

```
none when projid = 4136
```

If no SELECT_MSP or SELECT_VG parameter applies to a file, it will not be migrated.

The parameters are processed in the order that they appear in the policy. The first SELECT_MSP or SELECT_VG statement that applies to a given file is honored.

These parameters allow conditional expressions based on the value of a file tag. See "Customizing DMF" on page 186.

The root user on the DMF server can override the selection specified in these parameters through the use of dmput -V or with libdmfusr.so calls. If site–defined policies are in place, they may override these parameters.

There is no default.

See also:

- "Functions of policy Parameters" on page 365
- "Managed Filesystem Rules" on page 366

DCM MSP STORE_DIRECTORY policy Parameters

This section discusses the following:

- "`policy` Object Summary for a DCM MSP" on page 376
- "Automated Space-Management Parameters for a DCM MSP `STORE_DIRECTORY`" on page 376
- "File Weighting Parameters for a DCM MSP `STORE_DIRECTORY`" on page 378
- "VG Selection Parameters for a DCM MSP `STORE_DIRECTORY`" on page 380

See also "Functions of `policy` Parameters" on page 365.

`policy` Object Summary for a DCM MSP

Following is a summary of the potential parameters for the `policy` object for a DCM MSP:

```
policy
        TYPE
        DUALRESIDENCE_TARGET
        FREE_DUALRESIDENT_FIRST
        FREE_SPACE_DECREMENT
        FREE_SPACE_MINIMUM
        FREE_SPACE_TARGET
        SITE_SCRIPT
        CACHE_AGE_WEIGHT
        CACHE_SPACE_WEIGHT
        SELECT_LOWER_VG
```

For details, see:

- "Automated Space-Management Parameters for a DCM MSP `STORE_DIRECTORY`" on page 376
- "File Weighting Parameters for a DCM MSP `STORE_DIRECTORY`" on page 378
- "VG Selection Parameters for a DCM MSP `STORE_DIRECTORY`" on page 380

Automated Space-Management Parameters for a DCM MSP `STORE_DIRECTORY`

The following parameters control automated space management for a DCM MSP `STORE_DIRECTORY`:

Parameter	Description
DUALRESIDENCE_TARGET	Specifies the integer percentage of DCM MSP cache capacity that DMF maintains as a reserve of dual-resident files whose online space can be freed if free space reaches or falls below FREE_SPACE_MINIMUM. The dmdskmsp process tries to ensure that this percentage of the filesystem is copied to secondary storage, is currently being copied to secondary storage, or is free after it runs dmdskfree to make space available. This parameter is required for a DCM MSP; there is no default. (It does not apply to managed filesystems.)
FREE_DUALRESIDENT_FIRST	When set to ON, specifies that dmdskfree will first free dual-resident files before freeing files it must migrate. The default is OFF.
FREE_SPACE_DECREMENT	Specifies the integer percentage of filesystem space by which dmfsmon or dmdskmsp will decrement FREE_SPACE_MINIMUM if it cannot find enough files to migrate so that the value is reached. The decrement is applied until a value is found that can be achieved. If space later frees up, the FREE_SPACE_MINIMUM is reset to its original value. Valid values are in the range 1 through the value of FREE_SPACE_TARGET. The default is 2.
FREE_SPACE_MINIMUM	Specifies the minimum integer percentage of the total filesystem space that the DCM MSP tries to maintain as free. When the available free space reaches or falls below this threshold value, dmdskfree will begin to free dual-resident files and make non-dual-resident files dual-resident in order to meet the percentages set for FREE_SPACE_MINIMUM, FREE_SPACE_TARGET, and DUALRESIDENCE_TARGET. You should set FREE_SPACE_MINIMUM so that it is less than the values for FREE_SPACE_TARGET and DUALRESIDENCE_TARGET. This parameter is required; there is no default. For more details, see Chapter 11, "Automated Space Management" on page 509.

FREE_SPACE_TARGET	Specifies the integer percentage of total filesystem space that dmfsfree or dmdskfree tries to free if free space reaches or falls below the FREE_SPACE_MINIMUM threshold. You should set FREE_SPACE_TARGET so that it is less than DUALRESIDENCE_TARGET. This parameter is required; there is no default.
SITE_SCRIPT	Specifies the site-specific script to execute when dmfsfree, dmdskfree, or dmfsmon is run. If it returns a zero exit status, dmfsfree, dmdskfree, or dmfsmon continue their normal processing. If nonzero, they return immediately, using this value as their own exit status. See dmfsfree(8) or dmdskfree(8) for further details. This parameter is optional.

See also:

- "DCM MSP STORE_DIRECTORY Rules" on page 367
- "Functions of policy Parameters" on page 365

File Weighting Parameters for a DCM MSP STORE_DIRECTORY

The policy parameters for file weighting are as follows:

Parameter	Description
CACHE_AGE_WEIGHT	Specifies a floating-point constant and floating-point multiplier to use when calculating the weight given to a file's age, calculated as follows: *constant* + (*multiplier* * *file_age_in_days*) The default is a *constant* of 1 and a *multiplier* of 1. **Note:** This parameter refers to the copies in the cache, not the originals in the managed filesystem. The CACHE_AGE_WEIGHT parameter accepts an optional when clause, which contains a conditional expression. See "when Clause" on page 380.

Add a when clause to select which files should use these values. DMF checks each AGE_WEIGHT parameter in turn, in the order that they occur in the configuration file. If the when clause is present, DMF determines whether the file matches the criteria in the clause. If no when clause is present, a match is assumed. If the file matches the criteria, the file weight is calculated from the parameter values. If they do not match, the next instance of that parameter is examined.

You can configure a negative value to ensure that specific files are never automatically migrated. For example, you might want to set a minimum age for migration. The following parameter specifies that files that have been accessed or modified within 1 day are never automatically migrated:

```
CACHE_AGE_WEIGHT -1    0.0    when age <= 1
```

Note: DMF calculates the age weight and space weight separately. If either value is less than zero, the file is **not** automatically migrated and the file is **not** automatically freed. Otherwise, the two values are summed to form the file's weight.

CACHE_SPACE_WEIGHT

Specifies a floating-point constant and floating-point multiplier to use when calculating the weight given to a file's size, calculated as follows:

constant + (*multiplier* * *file_disk_space_in_bytes*)

The default is a *constant* of 0 and a *multiplier* of 0.

For a partial-state file, *file_disk_space_in_bytes* is the amount of space occupied by the file at the time of evaluation.

The CACHE_SPACE_WEIGHT parameter accepts an optional when clause, which contains a conditional expression. See "when Clause" on page 380.

Configure negative values to ensure that files are never automatically migrated. For example, you might want to set a minimum size for migration. The following parameter specifies that small files are never automatically migrated:

```
SPACE_WEIGHT -1    0    when space <= 4k
```

See also:

- "Functions of `policy` Parameters" on page 365
- "DCM MSP `STORE_DIRECTORY` Rules" on page 367

VG Selection Parameters for a DCM MSP `STORE_DIRECTORY`

The following parameter controls VG selection for a DCM MSP `STORE_DIRECTORY`:

Parameter	Description
`SELECT_LOWER_VG`	Defines the destinations of files that must be flushed to another MSP or VG. You can specify VGs, MSPs, and/or MGs, using the same syntax as `SELECT_MSP`/`SELECT_VG` in "MSP/VG Selection Parameters for a Managed Filesystem" on page 374 except that it must not include DCM MSPs. If no `SELECT_LOWER_VG` applies to a file, it will remain in the cache indefinitely. A large number of such files may impair the effectiveness of the DCM MSP or (in extreme cases) may cause the migration of more user files in the managed filesystem to fail. There is no default.

See also:

- "Functions of `policy` Parameters" on page 365
- "DCM MSP `STORE_DIRECTORY` Rules" on page 367

`when` Clause

The file weighting and MSP/VG selection parameters accept an optional `when` clause to restrict the set of files to which that parameter applies. It has the following form:

`when` *expression*

expression can include any of the following simple expressions:

Expression	Description
`age`	Specifies the number of days since last modification or last access of the file, whichever is more recent.
`gid`	Specifies the group ID or group name of the file.
`sitefn`	Invokes a site-defined policy function once for each file being considered, and is replaced by the return code of the function. This is only applicable to the `AGE_WEIGHT`, `SPACE_WEIGHT`, `SELECT_MSP`, and `SELECT_VG` parameters in a filesystem's `policy` stanza. For more information, see Appendix C, "Site-Defined Policy Subroutines and the `sitelib.so` Library" on page 735.
`projid`	Specifies a user-determined number associated with a file by the `dmprojid`(8) command, in the range `0`-`4294967294` providing that 32-bit project IDs are enabled in the filesystem; otherwise, the range is `0`-`65536`.

`sitetag`	Specifies a site-determined number associated with a file by the `dmtag`(1) command, in the range `0`–`4294967295`. For example:
	`sitetag = 27`
	`sitetag in (20-40, 5000, 4000000000)`

`size`	Specifies the logical size of the file, as shown by `ls -l`. By default, the unit of measure is bytes; see "Units of Measure" on page 275.
`softdeleted`	Specifies whether or not the file corresponding to a cached copy has been soft deleted; only applicable to the `CACHE_AGE_WEIGHT`, `CACHE_SPACE_WEIGHT`, and `SELECT_LOWER_VG` parameters in a DCM MSP `policy` stanza. Legal values are `false` and `true`.
`space`	Specifies the number of bytes that the file occupies on disk (always a multiple of the block size, which may be larger or smaller than the length of the file). There may be an optional trailing scaling character. Sparse files usually occupy less space on disk than the size of the file. For a partial-state (PAR) file, the value used is the

space that the file occupies on disk at the time of evaluation. The space that a file occupies on disk can change over time due to various effects, such as speculative preallocation. For this reason, `size` may be a better and more predictable metric to use in policies. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

For example, you might use the following line in a policy:

```
SELECT_VG none when space < 4096
```

Your intent would be to restrict files smaller than 4 Kbytes from migrating.

However, this line may actually allow files as small as 1 byte to be migrated, because while the amount of data in the file is 1 byte, it will take 1 block to hold that 1 byte. If your filesystem uses 4–Kbyte blocks, the space used by the file is 4096, and it does not match the policy line.

To ensure that files smaller than 4 Kbytes do not migrate, use the following line:

```
SELECT_VG none when space <= 4096
```

`uid` — Specifies the user ID or user name of the file.

Combine expressions by using `and`, `or`, and `()`.

Use the following operators to specify values:

```
=
!=
>
<
>=
<=
in
```

The following are examples of valid expressions:

```
space < 10m              (space used is less than 10 million bytes)
uid <= 123               (file's user ID is less than or equal to 123)
gid = 55                 (file's group ID is 55)
age >= 15                (file's age is greater than or equal to 15 days)
space > 1g               (space used is greater than 1 billion bytes)
uid in (chris, 10 82-110 200)
                         (file owner's user name is chris or
                         the file owner's UID is  10, in the range 82-110, or 200)
(gid = 55 or uid <= 123) and age < 5
                          (file's age is less than 5 days and its
                          group ID is 55 or its user ID is less than or equal to 123)
```

ranges Clause

If partial-state files are enabled on your host (meaning that you have the PARTIAL_STATE_FILES configuration file parameter set to ON, according to the information in the DMF release note), you can use the ranges clause to select ranges of a file. The AGE_WEIGHT and SPACE_WEIGHT parameters accept an optional ranges clause to restrict the ranges of a file for which a parameter applies. Example 7-25, page 387, shows an example of a policy that contains ranges clauses.

Note: The ranges clause is not valid with the CACHE_AGE_WEIGHT or CACHE_SPACE_WEIGHT parameters.

The clause has the following form, where *byteranges* is one or more byte ranges:

ranges *byteranges*

Each byte range consists of a set of numbers that indicate byte positions. (You can also use BOF or bof to indicate the first byte in the file and EOF or eof to indicate the last byte in the file.) Each byte range is separated by a comma and can have one of the following forms:

- A specification of two byte positions, where *first* specifies the first byte in the range and *last* specifies the last byte in the range:

  first:*last*

If unsigned, *first* and *last* count from the beginning of the file; if preceded by a minus sign (`-`), they count backwards from the end of the file.

The first byte in the file is byte `0` or `BOF` and the last byte is `-0` or `EOF`. Therefore, `BOF:EOF` and `0:-0` both define a range covering the entire file.

For example:

– `ranges 0:4095` specifies the first 4096 bytes of the file

– `ranges -4095:EOF` specifies the last 4096 bytes of the file

- A specification of the size of the range, starting at a given point, where *first* is a byte position as above and *size* is the number of bytes in the range, starting at *first*:

  first+size

  For example, the following indicates bytes 20 through 29:

  ```
  ranges 20+10
  ```

  If *size* is preceded by a minus sign, it specifies a range of *size* bytes ending at *first*. For example, the following indicates bytes 11 through 20:

  ```
  ranges 20+-10
  ```

- A specification of the size of the range only (without a colon or plus symbol), assumed to start at the end of file (when preceded by a minus sign) or beginning of file:

  -size
  size

  For example, the following specifies the last 20 bytes in the file:

  ```
  ranges -20
  ```

The *first*, *last*, or *size* values can be of the following forms:

- A hexadecimal number: `0x`*n*
- A decimal number with an optional trailing scaling character. The decimal number may include a decimal point (`.`) and exponent. The trailing scaling character may be one of those shown in "Units of Measure" on page 275.

Note: DMF may round byte ranges and join nearby ranges if necessary. If a range is given a negative weight, rounding may cause additional bytes to be ineligible for automated space management.

Do not use a `ranges` clause when partial-state files are disabled in DMF. Specifying many ranges for a file is discouraged, as it can cause the time and memory used by automated space management to grow. DMF has an upper limit on the number of regions that can exist within a file; this can sometimes cause a range to be given an effective lower weight than what was specified in the configuration file. This might happen if the file is already partial-state and the range with largest weight cannot be made offline (`OFL`) because that would create too many regions. If the file has too many regions to make the range offline, but it could be made offline at the same time as a range with lower weight, it will be given the lower weight. If more than one range in the middle of a file is not a candidate for automatic migration, the limit on the number of regions may make it impossible to automatically free other regions of the file.

`policy` Configuration Examples

This section discusses the following:

- "Automated Space-Management Example" on page 386
- "Automated Space-Management Using Ranges Example" on page 387
- "MSP/VG Selection Example" on page 388
- "`policy` Examples for Integrated Backups" on page 389

Automated Space-Management Example

Example 7-24 shows an example of a `policy` object to configure automated space management.

Example 7-24 `policy` Object for Automated Space Management

```
define  fs_space
        TYPE                    policy
        MIGRATION_TARGET        50
        FREE_SPACE_TARGET       10
        FREE_SPACE_MINIMUM      5
        FREE_DUALSTATE_FIRST    off

        AGE_WEIGHT 0    0.00    when age < 10
        AGE_WEIGHT 1    0.01    when age < 30
        AGE_WEIGHT 10   0.05    when age < 120
        AGE_WEIGHT 50   0.1

        SPACE_WEIGHT  0  0
enddef
```

In the above example:

- The `define` parameter must have a value that matches the value previously set in the `POLICIES` parameter of`filesystem` object.
- The automated space-management parameters specify that when only 5% (`FREE_SPACE_MINIMUM`) of the `fs_space` filesystem is free, DMF will try to migrate regular files until 10% (`FREE_SPACE_TARGET`) of the filesystem is free and 50% (`MIGRATION_TARGET`) of the filesystem is either free or has files that are dual-state. DMF will not free the space of existing dual-state files before beginning migration of regular files. (See "Automated Space-Management Parameters for a Managed Filesystem" on page 369.)

  DMF checks each `AGE_WEIGHT` parameter in turn, in the order that they occur in the configuration file. DMF checks the `when` clause to see if the file matches the criteria.
- File migration likelihood increases with the length of time since last access. Files that have been accessed or modified within the last 10 days have a weight of 0, making them the least likely to be migrated; files that have not been accessed or modified in 120 days or more have a far greater weight than all other files.

- The size of the file does not affect migration because all files have SPACE_WEIGHT of 0.

Automated Space-Management Using Ranges Example

Example 7-25 shows a policy object using ranges, which requires that partial-state files are enabled on the host (meaning that PARTIAL_STATE_FILES is set to ON and the appropriate kernel is installed, according to the information in the DMF release note).

Example 7-25 policy Object for Automated Space Management Using Ranges

```
define  fs2_space
        TYPE                    policy
        MIGRATION_TARGET        50
        FREE_SPACE_TARGET       10
        FREE_SPACE_MINIMUM      5
        FREE_DUALSTATE_FIRST    off

        AGE_WEIGHT -1.  0.00 ranges 0:4095 when uid=624
        AGE_WEIGHT -1   0    ranges 0:4095,-4095:EOF when uid=321
        AGE_WEIGHT 1    0.01     when age < 30
        AGE_WEIGHT 10   0.05     when age < 120
        AGE_WEIGHT 50   0.1

        SPACE_WEIGHT  0  0
enddef
```

The above example is similar to Example 7-24, page 386, with the following differences:

- If a file is owned by UID 624 and is 1004096 bytes long, the first 4096 bytes are given an AGE_WEIGHT of -1. The remaining 1000000 bytes are given an AGE_WEIGHT based on the age of the file; based on this weight, automated space management may select this file to be migrated. DMF migrates the entire file before changing its state to OFL, DUL, or PAR. Automated space management may also choose to put the last 1000000 bytes of the file offline based on the weight of that range; the first 4096 bytes will not be eligible for being put offline by automated space management.
- If a file is owned by UID 321, the first and last 4096 bytes of it will not be eligible for being put offline by automated space management, similar to the above situation.

- If a file is owned by UID 956, the policy in Example 7-25 would give the entire file an AGE_WEIGHT based on its age.

MSP/VG Selection Example

Example 7-26 defines a policy object for an MSP/VG.

Example 7-26 policy Object for an MSP/VG

```
define  fs_msp
        TYPE                    policy
        SELECT_MSP none         when space < 65536
        SELECT_MSP cart1 cart2  when gid = 22
        SELECT_MSP cart3        when space >= 10m
        SELECT_MSP cart1        when space >= 50m
        SELECT_VG cart2
enddef
```

In the above example:

- The define parameter must match the value that you set previously in the POLICIES parameter of the filesystem object. See "filesystem Object Parameters" on page 351.
- The special MSP name none means that files that are smaller than 65,536 bytes will never be migrated.
- The VG/MSP names (cart1, cart2, cart3) must match the names set in the LS_NAMES parameter (or else the MSP_NAMES parameter) of the dmdaemon object. The SELECT_MSP and SELECT_VG parameters are interchangeable, so both can be used in the same stanza.
- Any file with a group ID of 22 will be sent to both cart1 and cart2
- Smaller files will be sent to cart3 and larger files will be sent to cart1

Note: The order of the SELECT_MSP and SELECT_VG statements is important. The first SELECT statement that applies to the file is honored. For example, if the order of the statements above were reversed, a 10–million-byte file would be migrated to cart1, because the check for greater than or equal to 65,536 bytes would be done first, and it would be true.

- Any other file that does not meet the above criteria is sent to cart2.

`policy` Examples for Integrated Backups

Example 7-27, page 389 defines the `policy` objects for integrated backups.

Example 7-27 `policy` Objects for Integrated Backups

```
define backup_space_policy
        TYPE                    policy
        MIGRATION_TARGET        99
        FREE_SPACE_TARGET       98
        FREE_SPACE_MINIMUM      97
enddef

define backup_vg_policy
        TYPE                    policy
        SELECT_VG               none when projid = 4136
        SELECT_VG               bvg1a bvg1b
enddef
```

In the above example:

- The `define` parameter must match the value that you set previously in the `POLICIES` parameter of the `filesystem` object that sets `USE_AS_DUMP_DEST`. See "`filesystem` Object Parameters" on page 351.
- The automated space-management parameters specify that when 97% (`FREE_SPACE_MINIMUM`) of the filesystem is free, DMF will try to migrate regular files until 99% (`FREE_SPACE_TARGET`) of the filesystem is free and 97% (`MIGRATION_TARGET`) of the filesystem is either free or has files that are dual-state. DMF will not free the space of existing dual-state files before beginning migration of regular files. (See "Automated Space-Management Parameters for a Managed Filesystem" on page 369.)

  DMF checks each `AGE_WEIGHT` parameter in turn, in the order that they occur in the configuration file. DMF checks the `when` clause to see if the file matches the criteria.
- Files with project ID of `4136` (which are temporary files produced by the integrated-backup method) will never be migrated.

Note: The order of the `SELECT_VG` statements is important. The first `SELECT` statement that applies to a given file is honored.

- The VG names (bvg1a and bvg1b) must match the names set in the LS_NAMES parameter (or else the MSP_NAMES parameter) of the dmdaemon object.

fastmountcache Object

This section contains the following:

- "fastmountcache Object Name" on page 390
- "fastmountcache Object Summary" on page 390
- "fastmountcache Object Parameters" on page 390
- "fastmountcache Object Examples" on page 391

fastmountcache Object Name

The name of the fastmountcache object is chosen by the administrator and may contain uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

fastmountcache Object Summary

Following is a summary of the potential parameters for the fastmountcache object:

```
fastmountcache
        TYPE
        CACHE_MEMBERS
```

fastmountcache Object Parameters

The fastmountcache object defines the parameters for a fast-mount cache.

Parameter	Description
TYPE	Specifies fastmountcache (required name for this type of object). There is no default.

CACHE_MEMBERS	Specifies the MGs and independent VGs to be used as a fast-mount cache.

fastmountcache Object Examples

This section discusses the following examples:

- "fastmountcache with an MG" on page 391
- "fastmountcache with a Mix of Members" on page 391

fastmountcache with an MG

Example 7-28 fastmountcache with an MG

```
define  copan_fmc
        TYPE            fastmountcache
        CACHE_MEMBERS   mg_fmc
enddef
```

In the above example:

- The name of the fast-mount cache is copan_fmc.
- The VGs that will be used for the fast-mount cache are part of the mg_fmc MG.
- The volumegroup objects that are part of mg_fmc should use a RESERVED_VOLUMES setting of 1 to ensure proper rotation among the volumes.
- There must be a taskgroup object configured elsewhere in the configuration file to free the volumes in the fast-mount cache, and it must reference the taskgroup object named copan_fmc. See "taskgroup Object" on page 304.

fastmountcache with a Mix of Members

Example 7-29 fastmountcache with a Mix of Members

```
define  copan_fmc
        TYPE            fastmountcache
        CACHE_MEMBERS   vg_c00 mg_fmc
enddef
```

This example is similar to Example 7-28, page 391, except for the following:

- There is a single independent VG (`vg_c00`) and an MG (`mg_fmc`) that will be used for the fast-mount cache.
- The `vg_C00` VG should use the default setting of `RESERVED_VOLUMES` (which is `0`) because it should never participate in merging and no rotation is required.

Objects Related to an LS

Multiple objects are required to configure an LS. This section discusses the following:

- "`libraryserver` Object" on page 392
- "`drivegroup` Object" on page 395
- "`volumegroup` Object" on page 410
- "`migrategroup` Object" on page 426
- "`resourcescheduler` Object" on page 432
- "`resourcewatcher` Object" on page 433
- "`allocationgroup` Object" on page 434
- "Examples of Configuring an LS" on page 436
- "LS Tasks" on page 441
- "LS Database Records" on page 444

`libraryserver` Object

This section discusses the following:

- "`libraryserver` Object Name" on page 393
- "`libraryserver` Object Summary" on page 393
- "`libraryserver` Object Parameters" on page 393

libraryserver Object Name

The name of the libraryserver object is chosen by the administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

libraryserver Object Summary

Following is a summary of the potential parameters for the libraryserver object:

```
libraryserver
        TYPE
        CACHE_DIR
        CACHE_SPACE
        COMMAND
        COPAN_VSNS
        DISCONNECT_TIMEOUT
        DRIVE_GROUPS
        MAX_CACHE_FILE
        MESSAGE_LEVEL
        RUN_TASK
        TASK_GROUPS
        WATCHER
```

libraryserver Object Parameters

The libraryserver object, one for each library, has the following parameters:

Parameters	Description
TYPE	Specifies libraryserver (required name for this type of object). There is no default.
CACHE_DIR	Specifies the directory in which the VG stores chunks while merging them from sparse volumes. If you do not specify this parameter, DMF uses the value of TMP_DIR from the base object (see "base Object Parameters" on page 278). If you use Parallel DMF and specify CACHE_DIR, it must either be a CXFS filesystem or be in a CXFS filesystem. This directory must not be in a managed filesystem. By default, this value must not reside in the root filesystem (see the description of ADMDIR_IN_ROOTFS in "base Object Parameters" on

page 278). Do not change this parameter while DMF is running.

CACHE_SPACE

Specifies the amount of disk space that `dmatls` can use when merging chunks from sparse volumes. During merging, small chunks from sparse volumes are cached on disk before being written to a secondary storage. The default is `0`, which causes all files to be merged via sockets. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

The LS can merge volumes more efficiently if it can stage most of the files to disk.

Note: The zone size influences the required cache space. See `ZONE_SIZE` in "`volumegroup` Object" on page 410.

COMMAND

Specifies the binary file to execute in order to initiate the LS. This value must be `dmatls`.

COPAN_VSNS

(COPAN only) Specifies if the fourth character of the VSN indicates the RAID in the COPAN virtual tape library (VTL) or COPAN MAID that contains the volume. This specification applies for all VSNs in this LS. Specifying `ON` enables this feature; specifying `OFF` disables it. The default is `OFF`. Do not change this parameter while DMF is running.

DISCONNECT_TIMEOUT

Specifies the number of seconds after which the LS will consider a mover process to have exited if it cannot communicate with the process. Likewise, mover processes will use this value to determine if the LS has exited. The default is `180` seconds.

DRIVE_GROUPS

Names one or more `drivegroup` objects containing drives that the LS can use for mounting and unmounting volumes. There is no default.

The order of these names is significant. Where there are multiple copies of the data of migrated files, recalls will normally be directed to the first-named DG that is applicable. If more than one VG within a DG contains copies, the order of the names on `VOLUME_GROUPS`

	parameters is also relevant. Do not change this parameter while DMF is running. **Note:** See "Ensure that the Cache Copy is Recalled First" on page 116.
MAX_CACHE_FILE	Specifies the largest chunk (in bytes) that will be merged using the merge disk cache. Larger files are transferred directly via a socket from the read child to the write child. The default is 25% of the CACHE_SPACE value. Valid values are 0 through the value of CACHE_SPACE.
MESSAGE_LEVEL	Specifies the highest message level that will be written to the LS log, which includes messages from the LS's components. It must be an integer in the range 0–6; the higher the number, the more messages written to the log file. The default is 2.
RUN_TASK	See the description of RUN_TASK in "taskgroup Object Parameters" on page 312. Also see "Automated Maintenance Tasks" on page 175.
TASK_GROUPS	Names the taskgroup objects that contain tasks the LS should run. By default, no tasks are run.
WATCHER	Names the resource watcher that the LS should run. The default is no watcher. (A corresponding resourcewatcher object is required only if the default parameters are unacceptable. See "resourcewatcher Object Parameters" on page 434.)

See also "TMF Configuration Tasks" on page 505.

drivegroup Object

This section discusses the following:

- "drivegroup Object Name" on page 396
- "drivegroup Object Summary" on page 396
- "drivegroup Object Parameters" on page 397

drivegroup Object Name

The name of the drivegroup object is chosen by the administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

drivegroup Object Summary

Following is a summary of the potential parameters for the drivegroup object:

```
drivegroup
        TYPE
        AGGRESSIVE_HVFY
        BANDWIDTH_MULTIPLIER
        BLOCK_SIZE
        COMPRESSION_TYPE
        DRIVE_MAXIMUM
        DRIVE_SCHEDULER
        DRIVES_TO_DOWN
        FADV_SIZE_MAID
        LABEL_TYPE
        MAX_MS_RESTARTS
        MAX_PUT_CHILDREN
        MOUNT_BLOCKED_TIMEOUT
        MOUNT_SERVICE
        MOUNT_SERVICE_GROUP
        MOUNT_TIMEOUT
        MSG_DELAY
        MULTITAPE_NODES
        OV_ACCESS_MODES
        OV_INTERCHANGE_MODES
        POSITIONING
        POSITION_RETRY
        READ_ERR_MAXIMUM
        READ_ERR_MINIMUM
        READ_ERR_TIMEOUT
        READ_IDLE_DELAY
        REDIRECT_TIME_DG_DOWN
        REINSTATE_DRIVE_DELAY
        REINSTATE_VOLUME_DELAY
        REWIND_DELAY
        RUN_TASK
```

```
TASK_GROUPS
TMF_TMMNT_OPTIONS
VERIFY_POSITION
VOLUME_GROUPS
WRITE_CHECKSUM
```

drivegroup Object Parameters

The drivegroup object, one for each pool of interchangeable drives in a single library, has the following parameters:

Parameter	Description
TYPE	Specifies drivegroup (required name for this type of object). There is no default.
AGGRESSIVE_HVFY	Specifies whether or not DMF will set the hvfy flag on volumes in the VOL database for an expanded set of error conditions. You can set this parameter to ON or OFF. The default is OFF (disables). By default, DMF sets the hvfy flag for a volume in the VOL database when it determines that there have been an excessive number of errors while appending to the volume. (The hvfy flag prevents further writing to that volume; it must be manually cleared by using the dmvoladm command or the appropriate DMF Manager action.) If this parameter is enabled, DMF will also set the hvfy flag if there have been I/O or positioning errors in either of the following cases: • On the last two consecutive mounts, with two different drives involved • On three out of the last six mounts, with at least two different drives involved Errors that occur while recalling, merging, or migrating, are counted. DMF may also set hvfy on full tapes when this parameter is enabled.
BANDWIDTH_MULTIPLIER	*(OpenVault only)* Specifies a floating point number used to adjust the amount of bandwidth that the LS assumes a drive in this DG will use. The value is used when

scheduling drives, which allows the administrator to adjust for the affects of compression. The default is `1`, which means no compression. The minimum is `.1` and the maximum is `1000`. The `node` object parameters `HBA_BANDWIDTH` and `NODE_BANDWIDTH` are related to this parameter; see "`node` Object" on page 294.

`BLOCK_SIZE`

Specifies the maximum block size to use when writing from the beginning of a volume. The `blocksize` field in the database is updated with this value and is later used when reading or appending to a volume. For most storage devices, DMF supports block sizes ranging from `4096` – `2097152` bytes; for COPAN MAID and JBFS, DMF supports block sizes ranging from `131072` – `2097152` bytes. DMF uses direct I/O to tapes when possible. However, direct I/O cannot be used on some architectures if the block size is larger than `524288` bytes; in this case, DMF uses buffered I/O instead. DMF always uses buffered I/O for COPAN MAID devices. For more information, see "Device Block-Size Defaults and Bandwidth" on page 276. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

The default maximum size is dependent on your device configuration, show in "Device Block-Size Defaults and Bandwidth" on page 276.

`COMPRESSION_TYPE`

(COPAN MAID or JBFS only) Specifies the compression type and level to be used by the write child (`dmatwc`) mover process when writing from the beginning of the volume. The following values are accepted:

`snappy`	Uses the Snappy compression library.
`zlib`[:*level*]	Uses the `zlib` compression library with the specified compression level. See the `zlib`(3) man page for a description of the compression levels that

can be set; 1–9 are valid values. If you specify zlib without a value, 1 is the default level.

The compression level is set when an empty volume is first written and remains unchanged for that volume until it has been emptied and is rewritten. Compression and decompression are done by the mover process (dmatwc or dmatrc) when COPAN MAID or JBFS is used.

Note: If you specify COMPRESSION_TYPE, you must also specify compression for the OV_INTERCHANGE_MODES parameter (below); if you do not specify compression for OV_INTERCHANGE_MODES, the default is no compression.

If OV_INTERCHANGE_MODES specifies compression but COMPRESSION_TYPE is not specified, the default is snappy compression.

DRIVE_MAXIMUM

Specifies the maximum number of drives that the DG is allowed to use simultaneously. This may be more or less than the number of usable drives. The maximum is 100; the default is 100.

If you specify a negative number for this parameter, the DG will add it to the number of usable drives to derive the effective maximum. For example, if you specify -2 and there are 10 usable drives, then up to 8 of them can used by the VGs that belong to this DG. If a drive is then configured down (meaning that there are 9 usable), up to 7 drives can be used by the VGs.

Note: A given VG can have a lower maximum value if you specify the DRIVE_MAXIMUM parameter in its volumegroup stanza. See "volumegroup Object" on page 410.

DRIVE_SCHEDULER

Names the `resourcescheduler` objects that the DG should run for the scheduling of drives. The default is a resource scheduler of default type and parameters. For the defaults, see "`resourcescheduler` Object Parameters" on page 432.

DRIVES_TO_DOWN

Specifies an integer value that controls the number of "bad" drives the DG is allowed to try to configure down. When more than this number are down, whether due to the DG or to external influences such as the system administrator, the DG does not attempt to disable any more drives. The default of `0` prevents the DG from disabling any drives.

FADV_SIZE_MAID

(COPAN MAID only) Specifies when to call `posix_fadvise()` with advice `POSIX_FADV_DONTNEED` for COPAN MAID volumes. When a zone is ended, DMF calls `posix_fadvise()` provided that at least `FADV_SIZE_MAID` bytes have been written since the last call to `posix_fadvise()`. The minimum is `0`, which means that `posix_fadvise()` will never be called, and the maximum is `18446744073709551615`. The default is `100000000`. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

LABEL_TYPE

Specifies the label type used when writing volumes from the beginning. Possible values are:

- `al` (ANSI label)
- `nl` (no label)

  Note: `nl` is not recommended for data security reasons even though it might be slightly faster than the other values. `nl` is not allowed with COPAN MAID or JBFS.

- `sl` (standard label for IBM tapes)

The default is `al`.

MAX_MS_RESTARTS

Specifies the maximum number of times that DMF can attempt to restart the mounting service (TMF or OpenVault) without requiring administrator intervention. The default and recommended values are `1` for TMF and `0` for OpenVault.

MAX_PUT_CHILDREN

Specifies the maximum number of write child (`dmatwc`) processes that will be scheduled simultaneously for a DG. Each `dmatwc` process uses one drive. By specifying a process maximum, some drives can be reserved for recalls, provided that the total number of usable drives exceeds the process maximum.

You can specify values as follows:

- If the DG `DRIVE_MAXIMUM` is a positive value, that value is the maximum value that you can specify for the DG `MAX_PUT_CHILDREN`.
- If you specify a negative value for the DG `MAX_PUT_CHILDREN`, the DG will add it to the value of the DG `DRIVE_MAXIMUM` to derive the effective maximum number of processes for this DG. However, this can result in a situation in which files cannot be migrated even if there are usable drives, which can lead to filesystems filling.
- If you do not specify this parameter, then each VG uses its own value of `MAX_PUT_CHILDREN`.

Note: Even if you specify the DG `MAX_PUT_CHILDREN`, a given VG can have a lower maximum value if you specify the `MAX_PUT_CHILDREN` parameter in its `volumegroup` stanza. See "`volumegroup` Object" on page 410.

For example, if there are 6 usable drives available to the DG, the DG `DRIVE_MAXIMUM` is not specified, and the DG `MAX_PUT_CHILDREN` is configured to `-2`, then a maximum of 4 drives will be used for migrates. But if 4

drives are then configured down, at that point no drives can used for migrates.

Note: If you are using this `drivegroup` object for integrated backups, SGI recommends a value of `1` so that the dump data will be appended to a single tape if possible.

`MOUNT_BLOCKED_TIMEOUT` *(OpenVault only)* Specifies the maximum number of minutes to wait for a volume to be mounted when another process is using the drive. When OpenVault is the mounting service, DMF chooses which drive to use before starting a mover process. At the time it makes this choice, the drive is unused. During the small window between when this choice is made and when the mount is started, it is possible for a non-mover process to start using the drive. In that case, the mover process will block until either the drive becomes unused or *MOUNT_BLOCKED_TIMEOUT* minutes have passed, when the process will be told to exit, and the work will be scheduled for another drive. The default is `6`, the minimum is `4`, and the maximum is `35791394`.

`MOUNT_SERVICE` Specifies the mounting service. Possible values are `openvault` and `tmf`. You must use `openvault` for those DGs that contain drives on parallel data-mover nodes. The default is `openvault`.

Note: TMF is not supported on systems running the Red Hat Enterprise Linux operating system.

`MOUNT_SERVICE_GROUP` Specifies the name by which the DG's devices are known to the mounting service:

- OpenVault: use the OpenVault drive group name that is specified by the `ov_drivegroup` command.

Note: OpenVault and DMF each have a group of interchangeable devices known as a *drive group*. Despite their use of the same terminology, a DMF drive group is different from an OpenVault drive group, and need not use the same name.

- TMF: use the device group name that would be used with the `-g` option on the `tmmnt` command.

By default, the `drivegroup` object's name is used.

`MOUNT_TIMEOUT` Specifies the maximum number of minutes to wait for a volume to be mounted. When OpenVault is the mounting service, the smaller of *MOUNT_BLOCKED_TIMEOUT* and *MOUNT_TIMEOUT* has precedence when the mount is blocked because the drive is in use by another process; *MOUNT_TIMEOUT* applies to all other cases.

If a mount request waits for longer than this period of time, the DG attempts to stop and restart provided that the `MAX_MS_RESTARTS` parameter allows it. This is done in an attempt to force the hanging subsystem to resume normal operation or to fail solidly.

Do not make this value too restrictive, as any non-LS tape activity (including `xfsdump`) can legitimately delay a VG's volume mount, which could result in this timeout being exceeded. The maximum is `43920`. The default is `0`, which means infinity (that is, forever).

`MSG_DELAY` Specifies the number of seconds that all drives in the DG can be down before DMF sends an e-mail message to the administrator and logs an error message. The default is `0`, which means that as soon as DMF notices that the mounting service is up and all of the drives are configured down, it will e-mail a message. This delay also applies to email messages sent when no drives are available for migrates, which can happen if `MAX_PUT_CHILDREN` in the `drivegroup` stanza has a negative value.

MULTITAPE_NODES

(Parallel DMF and OpenVault only) Restricts the recall of a file that involves multiple tapes to one of the specified mover nodes. Without this restriction, if a given file was split across more than one tape, multiple mover nodes can simultaneously recall portions of it, which may cause a performance degradation.

DMF will choose an active and enabled node from this list. DMF will use only this node to recall multitape files until the node is no longer both active and enabled. Although you can modify the MULTITAPE_NODES list while DMF is running, it will not cause DMF to choose a new node. If you delete the MULTITAPE_NODES list, DMF will eventually stop restricting the recall capability.

The list may include the DMF server (if it acts as a mover node) as well as any parallel data-mover nodes. Use spaces to separate the node names.

For example, if you have four mover nodes (dmfserver, mover1, mover2, and mover3), and you want to restrict the recall of multiple tapes files to any one of them, you could enter the following:

```
MULTITAPE_NODES          mover1 mover2 mover3 dmfserver
```

Note: If all drives are down or busy on the chosen node, multitape recalls will wait until at least one drive is available.

If a multitape file is partial-state with multiple disjoint offline pieces, recall is not guaranteed to occur only on the chosen node.

OV_ACCESS_MODES

(OpenVault only) Specifies a list of names to be provided to OpenVault for the accessmode clause when mounting a volume. These are in addition to the names that DMF always specifies: variable and rewind. If you do not specify readwrite, DMF will provide readonly or readwrite, as appropriate. For more information about the possible values, see the

description of the `access` option in the `ov_mount`(8) man page.

`OV_INTERCHANGE_MODES` *(OpenVault only)* Specifies a list of mode values to be provided to OpenVault when writing a volume from the beginning. By default, this list is empty.

Most drives support a value of either `compression` or `nocompression`.

For example, to specify that you want data compressed, use:

```
OV_INTERCHANGE_MODES      compression
```

Compression and decompression are done by the mover process (`dmatwc` or `dmatrc`) when COPAN MAID or JBFS is used.

Note: If you use COPAN MAID or JBFS and specify `compression`, also see the `COMPRESSION_TYPE` parameter (above).

Some drives support additional values. For example, the `T10000C` drive also supports the additional values `T10000C`, `T10000B`, and `T10000A`. For example, if you have a mixture of T10000C and T10000B drives, you could use the following to tell the T10000C drives to write in compressed `T10000B` format so that both drives can then later read the same cartridges:

```
OV_INTERCHANGE_MODES      compression T10000B
```

For more information about the possible values, see the description of the `firstmount` option in the `ov_mount`(8) man page.

`POSITIONING` Specifies how the volume should be positioned. The values can be:

- `data`, which means:

  - When writing: use block ID seek capability to the zone if the block ID is known (the same as `direct`)
  - When reading: try to determine the block ID of the data being read and use the block ID seek capability to position there

- `direct`, which means use block ID seek capability to the zone if the block ID is known
- `skip`, which means use volume-mark skipping to the zone

The default depends on the type of drive, and is either `direct` or `data`. If data positioning is specified for a drive whose default is `direct`, the block ID is calculated by adding an estimate of the number of blocks from the start of the zone to the data being recalled and the block ID of the start of the zone. Not all drives use this format for block ID.

`POSITION_RETRY` Specifies the level of retry in the event of a failure during zone positioning. The values can be:

- `none`, which means there will be no retry
- `lazy`, which means the VG retries if a reasonably fast alternative means of positioning is available (default)
- `aggressive`, which means the VG can try more costly and time-consuming alternatives

If the VG is unable to position to a zone, all recalls for files with data in that zone are aborted by the VG (though not by DMF if a copy exists in another VG).

The default is `lazy`, to give the best overall recall time. If you are having trouble getting data from a volume, you might want to try `aggressive`.

`READ_ERR_MAXIMUM` Specifies the maximum number of I/O errors that will be tolerated when recalling a file. The legal range of values is `2`–`100000`. The default is `5000`. The value of

	READ_ERR_MAXIMUM should be greater than the value of READ_ERR_MINIMUM
	Note: READ_ERR_TIMEOUT, READ_ERR_MINIMUM, and READ_ERR_MAXIMUM together determine how many I/O errors will be tolerated when recalling a file. If the number of consecutive I/O errors is greater than READ_ERR_MAXIMUM, or if the number of consecutive I/O errors is greater than READ_ERR_MINIMUM and the elapsed number of seconds since the first error was seen is greater than READ_ERR_TIMEOUT, DMF will stop trying to recall the file from the current VG and will try to recall another copy (if one exists); otherwise, the recall will fail.
READ_ERR_MINIMUM	Specifies the number of errors (after the READ_ERR_TIMEOUT value has elapsed) that will cause DMF to stop trying to recall a file. The legal range of values is 1–100000. The default is 10. See the description of READ_ERR_MAXIMUM.
READ_ERR_TIMEOUT	Specifies the number of seconds that can elapse since the first I/O error was seen when recalling a file, after which DMF will stop trying to recall a file upon reaching the READ_ERR_MINIMUM value. The legal values for READ_ERR_TIMEOUT are 30 through 3600 seconds. The default is 600 seconds. See the description of READ_ERR_MAXIMUM.
READ_IDLE_DELAY	Specifies the number of seconds that an idle read child (dmatrc) can wait before being told to exit. If other DMF requests are waiting for a drive, the read child may be told to exit before *READ_IDLE_DELAY* seconds have passed. An individual volumegroup stanza may also specify READ_IDLE_DELAY; the value specified in the volumegroup stanza overrides the value in the drivegroup stanza (see "volumegroup Object Parameters" on page 411). If READ_IDLE_DELAY is not specified in either stanza, the default is 5 seconds.
REDIRECT_TIME_DG_DOWN	*(OpenVault only)* Specifies the number of seconds that DMF will wait before redirecting recalls; after all drives

have been down for *REDIRECT_TIME_DG_DOWN* seconds, DMF will start forwarding every recall to the VG that holds the secondary copy. (However, if a volume holding a secondary copy is in a tape that has been exported from the library, this value does not apply; in that case, DMF will wait for the drives to become available.) The default and maximum value is `2147483647`. The minimum value is `420`.

`REINSTATE_DRIVE_DELAY` Specifies the number of minutes after which a drive that was configured down by the DG will be automatically reinstated and made available for use again. A value of `0` means it should be left disabled indefinitely. The default is `1440` (one day).

`REINSTATE_VOLUME_DELAY` Specifies the number of minutes after which a volume that had its `hlock` flag set by DMF will be automatically reinstated and made available for use again. A value of `0` means the volume should be left disabled indefinitely. The default is `1440` (one day).

`REWIND_DELAY` Specifies the number of seconds that an idle read child (`dmatrc`) can wait before rewinding. If other DMF requests are waiting for a drive, the read child may rewind before *REWIND_DELAY* seconds have passed. The maximum is the value of `READ_IDLE_DELAY`. If `READ_IDLE_DELAY` is not specified, the maximum is the default value of `READ_IDLE_DELAY`. An individual `volumegroup` stanza may also specify `REWIND_DELAY`; the value specified in the `volumegroup` stanza overrides the value in the `drivegroup` stanza (see "`volumegroup` Object Parameters" on page 411). If `REWIND_DELAY` is not specified in either stanza, then the default is the minimum of:

`{2, READ_IDLE_DELAY/2}`

If an idle read child must rewind the volume before the drive can be used to service other DMF requests, that will delay the servicing of those requests; therefore you should use caution when increasing this parameter.

RUN_TASK	See the description of RUN_TASK in "taskgroup Object Parameters" on page 312. Also see "Automated Maintenance Tasks" on page 175.
TASK_GROUPS	Names the taskgroup objects that contain tasks the DG should run. By default, no tasks are run.
TMF_TMMNT_OPTIONS	*(TMF only)* Specifies command options that should be added to the tmmnt command when mounting a tape. DMF uses the -Z option to tmmnt to ignore options controlling block size and label parameters. Use the BLOCK_SIZE and LABEL_TYPE DG parameters instead. There is no need for a -g option here. If it is provided, it must match the value of the MOUNT_SERVICE_GROUP parameter. To request compression, use -i. Options that are ignored are -a, -b, -c, -D, -f, -F, -l, -L, -n, -o, -O, -p, -P, -q, -R, -t, -T, -U, -v, -V, -w, -x, and -X.
VERIFY_POSITION	Specifies whether the LS write child should (prior to writing) verify that the volume is correctly positioned and that the volume was properly terminated by the last use. You can set this parameter to ON or OFF. The default is ON (verify).
VOLUME_GROUPS	Names the volumegroup objects containing volumes that can be mounted on any of the drives within this DG. There is no default. The order of these names is significant. Where there are multiple copies of the data of migrated files, recalls will normally be directed to the first-named VG that is applicable. A given volumegroup object can be listed in only one drivegroup object. Do not change this parameter while DMF is running. **Note:** See "Ensure that the Cache Copy is Recalled First" on page 116.
WRITE_CHECKSUM	Specifies if blocks should be checksummed before writing. If a block has a checksum, it is verified when

read. You can set this parameter to ON or OFF. The default is ON.

See also Procedure 9-1 in "Configure OpenVault for a Drive Group" on page 502.

volumegroup Object

This section discusses the following:

- "volumegroup Object Name" on page 410
- "volumegroup Object Summary" on page 410
- "volumegroup Object Parameters" on page 411
- "volumegroup Object Example with an AG" on page 423
- "volumegroup Object Example Overriding drivegroup Settings" on page 426

volumegroup Object Name

The name of the volumegroup object is chosen by the administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

DMF supports up to 1024 MSPs, VGs and/or MGs.

volumegroup Object Summary

Following is a summary of the potential parameters for the volumegroup object:

```
volumegroup
        TYPE
        ALLOCATION_GROUP
        ALLOCATION_MAXIMUM
        ALLOCATION_MINIMUM
        CHECKSUM_TYPE
        DRIVE_MAXIMUM
        FORWARD_RECALLS
        FULL_ZONE
        GET_WAIT_TIME
        HFREE_TIME
        IMPORT_ONLY
```

```
KILL_WC_IF_DEMAND
LOGICAL_BLOCK_PROTECTION
MAX_CHUNK_SIZE
MAX_IDLE_PUT_CHILDREN
MAX_PUT_CHILDREN
MERGE_CUTOFF
MERGE_THRESHOLD
MIN_VOLUMES
PUT_IDLE_DELAY
PUTS_TIME
READ_IDLE_DELAY
READ_TIME
RESERVED_VOLUMES
REWIND_DELAY
RUN_TASK
TASK_GROUPS
TIMEOUT_FLUSH
VOL_MSG_TIME
ZONE_SIZE
```

`volumegroup` Object Parameters

There must be a `volumegroup` object for each pool of volumes of the same type. It must be usable on the drives of the associated DG and capable of holding at most one copy of user files.

Note: The `run_tape_merge.sh` and `run_merge_stop.sh` tasks and their associated parameters can be specified in the `volumegroup` object.

A `volumegroup` object has the following parameters:

Parameter	Description
`TYPE`	Specifies `volumegroup` (required name for this type of object). There is no default.
`ALLOCATION_GROUP`	Names the allocation group (AG) that serves as a pool of additional volumes for this VG. The name may contain up to 8 uppercase or lowercase alphanumeric characters or underscores; it cannot begin with an underscore, contain any white space, or be the same as

a volumegroup object name. See "allocationgroup Object" on page 434 and "volumegroup Object Example with an AG" on page 423.

ALLOCATION_MAXIMUM — Specifies the maximum size in number of volumes to which a VG can grow by borrowing volumes from its AG. The minimum is 0, the maximum and default are 2147483647. If the VG already contains ALLOCATION_MAXIMUM or more volumes, no additional volumes are borrowed from the AG. If no AG is defined, this parameter is meaningless.

ALLOCATION_MINIMUM — Specifies the minimum size in number of volumes to which a VG can shrink by returning volumes to its AG. The minimum and default are 0 and the maximum is the value of ALLOCATION_MAXIMUM. If the VG already contains ALLOCATION_MINIMUM or fewer volumes, no additional volumes are returned to the AG. If no AG is defined, this parameter is meaningless.

KILL_WC_IF_DEMAND — Specifies whether or not an idle write child will be terminated immediately if there is demand for the drive. By default, one write child per VG is allowed to stay alive for a short period of time, even if there is demand for a drive, and even if the MAX_IDLE_PUT_CHILDREN value is 0. (If the MAX_IDLE_PUT_CHILDREN parameter is set to a nonzero value, that number of write children will be allowed to stay alive after the interim time has passed, so long as the drive is not in demand.) You can set this parameter to ON or OFF. The default is OFF.

CHECKSUM_TYPE — Specifies the type of checksum algorithm to use when writing new tapes. Possible values are:

- crc32c specifies the CRC32C checksum algorithm, which is required for the logical block protection feature (see LOGICAL_BLOCK_PROTECTION below). You must use the default setting for WRITE_CHECKSUM (ON) when specifying this value (see "drivegroup Object Parameters" on page 397). Any parallel data-mover nodes that use tapes in this VG must be running a version of the DMF software

that supports this checksum type. `crc32c` is the default for JBFS volumes.

Note: Using `crc32c` may set the actual size of the blocks written to the tape to be 4 bytes fewer than the *BLOCK_SIZE* value (see "`drivegroup` Object Parameters" on page 397).

- `legacy` specifies the original checksum algorithm used with DMF. `legacy` is the default for all other volume types.

`DRIVE_MAXIMUM` — Specifies the maximum number of drives that this VG is allowed to use simultaneously. The maximum value for this parameter is `100`; the default is the DG `DRIVE_MAXIMUM` (see "`drivegroup` Object Parameters" on page 397).

If you specify a negative value for the VG `DRIVE_MAXIMUM`, it will be added to the value used for the DG `DRIVE_MAXIMUM` and the sum will be the effective maximum for this VG.

However, the number of drives actually used is the least of the following:

- The VG `DRIVE_MAXIMUM` effective value
- The DG `DRIVE_MAXIMUM` effective value
- The number of usable drives

For example, suppose that the VG `DRIVE_MAXIMUM` is `-1` and the DG `DRIVE_MAXIMUM` is `-2`. If there are 10 usable drives, then up to 7 of them will be available to this VG. If a drive is then configured down, at that point up to 6 will be available to this VG.

`FORWARD_RECALLS` — Specifies whether or not a recall should be directed to another VG or MSP if the volume required for the recall is unavailable because it is being written to. If no other VG or MSP can satisfy the request, it will be handled by this VG. Use of this parameter may cause additional

volume mounts because the decision whether to forward a recall depends on whether the volume is being written at the time the recall request is received. You can set this parameter to ON or OFF. The default is OFF.

For more information about the use of this parameter in conjunction with tapes that have been exported from a library, see "OpenVault and Out-of-Library Tapes" on page 185. For more information about recalling from volumes being written, see GET_WAIT_TIME in "volumegroup Object Parameters" on page 411.

FULL_ZONE

Limits when requests are flushed to a mounted volume that is not writing to a zone. This applies only during the short period of time when one write child per VG is allowed to say alive, even if there is demand for a drive, and even if the MAX_IDLE_PUT_CHILDREN value is 0; after this time expires, FULL_ZONE is not considered. If FULL_ZONE is set ON, requests are flushed only when a full zone has accumulated; otherwise, requests are flushed as they arrive. (Note that requests are always flushed in the following circumstances: if the write child starts, if they are queued when a zone is completed, if TIMEOUT_FLUSH time has passed, or if a dmdidle command has been issued. The FULL_ZONE setting does not impact any of these actions). You can set this parameter to ON or OFF. The default is OFF.

GET_WAIT_TIME

Limits the amount of time (in seconds) that DMF will continue writing to a volume after receiving a recall request for that volume. In the case where there is queued writing work, the process writing to a volume will stop writing after it has finished a zone, providing a recall request for the volume has been queued for at least GET_WAIT_TIME seconds. If socket merges are taking place, it is possible that a few additional chunks may be written after the end of the zone before writing to the volume is stopped (depending on the size of the chunks). The legal range of values is 600 - 2147483647; the default is 2147483647.

To minimize extra volume mounts and the number of partial volumes created, do not make this value too small. Other requests may be queued before the recall request, therefore it may not proceed immediately after DMF stops writing to the volume. Also see FORWARD_RECALLS for more information about recalling from a volume that is being written.

HFREE_TIME

Specifies the minimum number of seconds that a volume no longer containing valid data must remain unused before the VG overwrites it. The default is 172800 seconds (2 days) and the minimum is 0.

Note: If you are using this volumegroup object for integrated backups, SGI recommends that you set HFREE_TIME to 1209600 (2 weeks). See "Consideration for Integrated Backups: Restored Database May Point to Freed Space" on page 160.

When an LS removes all data from a volume, it sets the hfree (hold free) flag bit in the volume's VOL record in the LS database to prevent that volume from being immediately reused. The next time that the LS scans the database for available volumes that can be assigned to volume groups after HFREE_TIME seconds have passed, the LS clears the hfree flag, allowing the volume to be rewritten. If HFREE_TIME is set to 0, the LS will never clear hfree, so an unused volume will not be reused until you clear its hfree flag manually. For a description of how to set and clear the hfree flag manually, see the dmvoladm man page.

IMPORT_ONLY

Specifies if the VG is used for importing only. You can set this parameter to ON or OFF. The default is OFF. Set this parameter ON when the data in the VG is being migrated to another VG, perhaps as part of a media hardware upgrade. The daemon will not accept dmput(1), dmmove(8), or dmarchive(1) requests that specify a VG with this parameter enabled.

When the DMF daemon performs a complete file recall from an import-only VG and all other DMF copies also reside in import-only MSPs or VGs, it makes the file a regular file (rather than a dual-state file) and it soft-deletes the VG's copy of the file.

Note: An import-only VG should never be a member group of a `migrategroup` stanza.

`LOGICAL_BLOCK_PROTECTION`

Specifies whether logical block protection should be turned on when reading and writing tapes. This feature applies only to Oracle's StorageTek T10000C and later models that support data integrity validation (see the drive manufacturer's specifications for the required level of firmware). You can set this parameter to `ON` or `OFF`. The default is `OFF`.

If `ON` is specified, you must do all of the following:

- Use a supported drive
- Set `CHECKSUM_TYPE` to `crc32c` in the `volumegroup` stanza
- Use the default setting for `WRITE_CHECKSUM` (`ON`) in the `drivegroup` stanza (see "`drivegroup` Object Parameters" on page 397)

Otherwise, an error message is logged.

`MAX_CHUNK_SIZE`

Specifies the size of the chunk into which the VG should break up large files as it writes data to secondary storage. If a file is larger than this size, it is broken up into pieces of the specified size. Depending on other activity, more than one write child may be used to write the data to secondary storage. If `MAX_CHUNK_SIZE` is `0` (the default), the VG breaks a file into chunks only when an end-of-volume is reached. By default, the unit of measure is bytes; see "Units of Measure" on page 275. If you set this parameter in a Parallel DMF configuration where drives

in the drivegroup are usable from more than one node, then you should also set MULTITAPE_NODES; see "drivegroup Object Parameters" on page 397.

MAX_IDLE_PUT_CHILDREN

Specifies the maximum number of idle write child (dmatwc) processes that will be allowed simultaneously for a VG. The maximum is the value of MAX_PUT_CHILDREN for the VG. The minimum and default are 0. If you specify a non-zero value, idle dmatwc processes will be allowed to stay alive, with a volume mounted, for a maximum of PUT_IDLE_DELAY seconds. During this time, if sufficient migrates arrive to fill a zone, they can be given to an idle dmatwc process.

Note: If the drive is needed for other work, there may be additional delay caused by the time needed to rewind and unmount the tape associated with the idle process. There may be times when the number of idle write children will exceed this value; for example, if socket merges are occurring or immediately after a dmatwc process completes a zone. If you configure MAX_IDLE_PUT_CHILDREN, you must choose its value and the value of PUT_IDLE_DELAY with the following in mind:

- OpenVault: DMF can take several minutes to respond when a drive is needed for some purpose other than a recall or migrate (for example, for a dmatsnf or xfsdump request)
- TMF: DMF will not notice that a drive is needed for another purpose

---

MAX_PUT_CHILDREN

Specifies the maximum number of write child (dmatwc) processes that will be scheduled simultaneously for this VG.

The maximum is the DG MAX_PUT_CHILDREN effective value (if specified) or else the DG DRIVE_MAXIMUM effective value. The minimum is 1. (A negative value is invalid.)

The default is the DG MAX_PUT_CHILDREN value, if specified. If you do not configure the DG MAX_PUT_CHILDREN, the default is the same as the VG DRIVE_MAXIMUM value.

However, the number of process actually scheduled is the least of the following:

- The VG MAX_PUT_CHILDREN effective value
- The DG MAX_PUT_CHILDREN effective value
- The VG DRIVE_MAXIMUM effective value
- The number of usable drives

Note: If you are using this volumegroup object for integrated backups, SGI recommends a value of 1 so that the dump data will be appended to a single tape if possible.

Also see "Configure Appropriately for SGI 400 VTL or COPAN MAID Shelves" on page 109.

MERGE_CUTOFF

Specifies a limit at which the VG will stop scheduling volumes for merging. This number refers to the sum of the active and queued mover children that are generated from gets, puts, and merges (that is, the dmatwc and dmatrc processes). The default value for this option is the value used by the volumegroup object for DRIVE_MAXIMUM. This means that if sparse volumes are available, the VG will create DRIVE_MAXIMUM number of children, thus using resources efficiently. However, if any recall requests arrive for that VG, they will be started before new merges. Setting this number below DRIVE_MAXIMUM reserves some volumes for recalls at the expense of merge efficiency. Setting this number above DRIVE_MAXIMUM increases the priority of merges relative to recalls. The minimum value is 2.

MERGE_THRESHOLD

Specifies the integer percentage of active data on a volume less than which DMF will consider a volume to

be sparse and allow merging; a value of `0` prohibits merging. This parameter overrides the `THRESHOLD` parameter (defined in a `taskgroup` stanza) for this VG, which allows each VG to have a different sparse volume threshold. If a VG is part of a fast-mount cache, you must set this parameter to `0`. The default is the `THRESHOLD` parameter; see "`taskgroup` Object Parameters" on page 312.

`MIN_VOLUMES` — Specifies the minimum number of unused volumes that can exist in the LS database for this VG without operator notification. If the number of unused volumes falls below `MIN_VOLUMES`, the operator is asked to free-up new volumes. The minimum is `0`, the maximum is `2147483647`, and the default is `10`. If a VG has an AG configured, `MIN_VOLUMES` is applied to the sum of the number of unused volumes in the VG and in its AG subject to any `ALLOCATION_MAXIMUM` restrictions.

`PUT_IDLE_DELAY` — Specifies the number of seconds that an idle write child (`dmatwc`) process will be allowed to stay alive. The default value is `30` seconds.

Note: If you configure `PUT_IDLE_DELAY`, you should also specify `MAX_IDLE_PUT_CHILDREN` and consider the implications of these values on other work that may be needed for the drive. See the **Note** under `MAX_IDLE_PUT_CHILDREN`.

`PUTS_TIME` — Specifies the minimum number of seconds that a VG waits after it has requested a drive for a write child before it tells a lower priority child to go away. The default is `3600` seconds.

`READ_IDLE_DELAY` — Specifies the number of seconds that an idle read child (`dmatrc`) can wait before being told to exit. If other DMF requests are waiting for a drive, the read child may be told to exit before *READ_IDLE_DELAY* seconds have passed. The `drivegroup` stanza may also specify `READ_IDLE_DELAY`; the value specified in the `volumegroup` stanza overrides the value in the `drivegroup` stanza. The default value of

READ_IDLE_DELAY is the value that is specified in the drivegroup stanza (see "drivegroup Object Parameters" on page 397).

READ_TIME — Specifies the interval (in seconds) after which the VG will evaluate whether a read child should be asked to go away (even if it is in the middle of recalling a file) so that a higher priority child can be started. If READ_TIME is 0, the VG will not do this evaluation. The default is 0.

RESERVED_VOLUMES — Defines the number of remaining empty volumes that will cause the VG to stop accepting migration requests:

- If merging is required for the VG, the reserved volumes will be used for merging. Reserving volumes prevents the situation where all volumes become full and there are no volumes available for merging.
- If the VG is in an MG, the requests will be sent to another VG in the MG. This enables an MG to avoid queuing requests to a full VG when there are available volumes in another VG.

When the number of empty, writable volumes drops below this threshold, the VG will fail migrations. If the migration target was not in an MG, dmfsfree or dmmigrate may try repeatedly to migrate the file and DMF logs may grow to a very large size.

Set RESERVED_VOLUMES as follows:

Setting	Circumstance
0	For a VG that is an independent member of a fast-mount cache (that is, this VG is listed in the CACHE_MEMBERS parameter, see "fastmountcache Object Parameters" on page 390). This is the default.

1	For every VG that is part of an MG in a fast-mount cache configuration (that is, the MG is listed in CACHE_MEMBERS)
1 or more	Either of the following: • For all VGs except the last in an MG that is not part of a fast-mount cache and that has a ROTATION_STRATEGY of SEQUENTIAL • For a VG on a COPAN shelf that is used as secondary storage

Note: If you set this parameter is set to a non-zero value, you should set EXPORT_METRICS to ON (see "base Object" on page 277).

REWIND_DELAY

Specifies the number of seconds that an idle read child (dmatrc) can wait before rewinding. If other DMF requests are waiting for a drive, the read child may rewind before REWIND_DELAY seconds have passed. The maximum is the value of READ_IDLE_DELAY. REWIND_DELAY may also be specified in the drivegroup stanza; if REWIND_DELAY is specified in the drivegroup stanza but not in the volumegroup stanza, then the drivegroup value is used (see "drivegroup Object Parameters" on page 397). If REWIND_DELAY is not specified in either stanza, then the default is the minimum of:

{2, *READ_IDLE_DELAY*/2}

If an idle read child must rewind the volume before the drive can be used to service other DMF requests, that will delay the servicing of those requests; therefore you should use caution when increasing this parameter.

RUN_TASK

See the description of RUN_TASK in "taskgroup Object Parameters" on page 312. Also see "Automated Maintenance Tasks" on page 175.

TASK_GROUPS	Names the `taskgroup` objects that contain tasks the VG should run. By default, no tasks are run. The only defined tasks that can be run in a VG `taskgroup` are `run_tape_merge.sh` and `run_merge_stop.sh`.
TIMEOUT_FLUSH	Specifies the number of minutes after which the VG will flush files to secondary storage, even if the flush does not produce a full volume zone. The minimum is `0` (that is, do not flush based on time), the maximum is `35791394`, and the default is `120` minutes. See also "Appropriately Force Small Files to Migrate" on page 107.
VOL_MSG_TIME	Specifies the minimum interval (in seconds) between operator notifications for low–volume and no-volume conditions for this VG. DMF will send at most one message for each occurrence of a low-volume condition and one message for each occurrence of a no-volume condition. Volumes that are actively being used are not considered available, and so a VG may fall below the low-volume or no-volume threshold, and then as the volumes are no longer being used, it may rise above the threshold. This can trigger frequent notifications when the VG is close to the threshold. You can use this parameter to reduce the number of messages sent. Additional notifications may be sent when the VG has no writable volumes at all or when the number of empty + partial volumes falls below the threshold. The default value is `86400` seconds (24 hours), the minimum value is `0`, and the maximum value is `2147483647`. Also see the `MIN_VOLUMES` parameter in "`drivegroup` Object" on page 395

ZONE_SIZE	Specifies approximately how much data the write child should put in a zone. The write child adds files and chunks to a zone until the data written equals or exceeds this value, at which time it writes a volume mark and updates the database. The VG also uses zone size to determine when to start write children and the number of write children to start. The minimum is 1000000 bytes and the maximum is 1000000000000000, and the default is 50000000 bytes (or 50m). By default, the unit of measure is bytes; see "Units of Measure" on page 275. For more information about zone size, also see "Media Concepts" on page 534 **Note:** It is critical that the zone size is appropriate for the media speed and average data compression rate at your site. A value that is too small can cause poor write performance because a volume mark is written at the end of each zone; a value that is too large can reduce parallelism when migrating files. See "Improve Drive Performance with an Appropriate VG Zone Size" on page 105. The zone size influences the required cache space. The value for the CACHE_SPACE parameter should be at least twice the value used for ZONE_SIZE. Increasing the ZONE_SIZE value without also increasing CACHE_SPACE could cause volume merging to become inefficient. Merges could have problems if the ZONE_SIZE value is larger than the CACHE_SPACE value. For more information about CACHE_SPACE, see "libraryserver Object Parameters" on page 393. See also "Appropriately Force Small Files to Migrate" on page 107

volumegroup Object Example with an AG

You can include an optional AG to provide a logical pool of additional volumes that are available to multiple VGs. These volumes will automatically be transferred to a

given VG as they are needed. When free, they can be immediately returned to the AG, making them eligible for use by another VG. This movement of volumes in and out of the AG is subject to the restrictions imposed by HFREE_TIME, ALLOCATION_MAXIMUM and ALLOCATION_MINIMUM.

To identify the AG for given VG, include the ALLOCATION_GROUP parameter within its volumegroup object. Normally, you will use one AG to serve multiple VGs by including the same ALLOCATION_GROUP parameter value in the definition of multiple volumegroup objects.

When you add a group of volumes to the VOL database, you can explicitly assign them to a specific AG by using the dmvoladm (8) command. If a VG has free volumes at the time when you add an AG to its stanza, those free volumes will automatically move into the AG (subject to the restrictions imposed by the configuration parameters ALLOCATION_MAXIMUM and ALLOCATION_MINIMUM).

Note: Any volume that is assigned to an AG must be usable by any of the VGs that will use the AG. That is, you must ensure that volumes assigned to the AG are mountable on drives in the same DG as any VG that references the AG.

An ALLOCATION_GROUP name cannot be the same as a volumegroup object name.

A VG must define the ALLOCATION_GROUP option in order to use an AG

Example 7-30 volumegroup example with an AG

Do the following:

1. *(Optional)* Assign the volumes to the AG when you add them to the VOL database by using the dmvoladm(8) command:

   # **dmvoladm -l *LS* -c "create *VSNs* volgrp *AG*"**

   where:

Option	Description
LS	Specifies the name of the LS that owns the VOL records
VSNs	Specifies one VSN or a range of VSNs separated by the hyphen (-) character
AG	Specifies the AG to which *VSNs* will be assigned

Note: The **volgrp** field keyword specifies either a VG name or an AG name.

For example, the following line would assign volumes with the VSNs X04000 through X04010 to the AG ag1:

```
# dmvoladm -l ls1 -c "create X04000-X04010 volgrp ag1"
```

For more information, see the dmvoladm(8) man page.

2. Include the AG in the DMF configuration file according to the information in "volumegroup Object" on page 410.

   a. Add the ALLOCATION_GROUP parameter to the volumegroup stanza for each VG that should use the pool of volumes. For example, the following extract shows that ag1 will supply volumes to both vg1 and vg2:

```
define  vg1
                TYPE                    volumegroup
                ALLOCATION_GROUP        ag1
enddef

define  vg2
                TYPE                    volumegroup
                ALLOCATION_GROUP        ag1
enddef
```

   Note: An AG only requires an allocationgroup stanza if you wish to change its VOL_MSG_TIME setting.

   b. Define the ALLOCATION_MAXIMUM and ALLOCATION_MINIMUM parameters as needed in the volumegroup stanza to restrict the size of the VG.

3. (Optional) To change the minimum interval between operator notifications for low–volume and no-volume conditions for the AG to 48 hours (172800 seconds), you could add the optional allocationgroup stanza:

```
define  ag1
                TYPE                    allocationgroup
                VOL_MSG_TIME            172800
enddef
```

For a more complete example, see "Example: LS with a Resource Watcher, Two DGs, and an AG" on page 436.

`volumegroup` Object Example Overriding `drivegroup` Settings

The following excerpts illustrate how setting values in the `volumegroup` stanza override the `drivegroup` settings. Suppose you have the following:

```
define dg1
                TYPE                    drivegroup
                VOLUME_GROUPS           vg1 vg2
                MOUNT_SERVICE_GROUP     dg_c00
enddef

define vg1
                TYPE                    volumegroup
                READ_IDLE_DELAY         0
enddef

define vg2
                TYPE                    volumegroup
enddef
```

The default for `REWIND_DELAY` depends upon the setting for `READ_IDLE_DELAY`:

- For `vg1`, `READ_IDLE_DELAY` is set to `0`. Therefore, the default value used for `REWIND_DELAY` is calculated by using the minimum of `2` and *`READ_IDLE_DELAY`*`/2` (in this case `0/2`), making the default `0`.
- For `vg2`, `READ_IDLE_DELAY` is not explicitly set. Therefore, the default value used for `REWIND_DELAY` is determined by using the default value of `READ_IDLE_DELAY` (`5`) in calculating the minimum of `2` and *`READ_IDLE_DELAY`*`/2` (in this case `5/2`), making the default `2`.

`migrategroup` Object

This section discusses the following:

- "`migrategroup` Object Name" on page 427

- "migrategroup Object Summary" on page 427
- "migrategroup Object Parameters" on page 427
- "migrategroup Object Example with Multiple MGs" on page 431
- "Single migrategroup Object Example Using the ROUND_ROBIN_BY_BYTES Strategy" on page 431
- "migrategroup Object Example Using the ROUND_ROBIN_BY_FILES Strategy" on page 431

migrategroup Object Name

The name of the migrategroup object is chosen by the administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

DMF supports up to 1024 MSPs, VGs and/or MGs.

migrategroup Object Summary

Following is a summary of the potential parameters for the migrategroup object:

```
migrategroup
        TYPE
        GROUP_MEMBERS
        MULTIPLIER
        ROTATION_STRATEGY
```

migrategroup Object Parameters

Warning: Never add, delete, or change the order of migrategroup stanzas while DMF is running; making changes of this type can result in data corruption or data loss.

There can be a migrategroup object for each set of MSPs/VGs that you want to treat as a single migration target.

A migrategroup object has the following parameters:

Parameter	Description
TYPE	Specifies migrategroup (required name for this type of object). There is no default.
GROUP_MEMBERS	Specifies the list of MSPs/VGs that are members of this MG. Each migration request will result in exactly one copy being made to a member MSP/VG. The order of the group members is significant if you use a ROTATION_STRATEGY of SEQUENTIAL. The members must have their own volumegroup or msp stanzas in the configuration file. See: • "volumegroup Object" on page 410 • "msp Object for a Cloud MSP" on page 448 • "msp Object for an FTP MSP" on page 455 • "msp Object for a Disk MSP" on page 466 • "msp Object for a DCM MSP" on page 459 Do not change this parameter while DMF is running. Do not include an import-only MSP/VG.
MULTIPLIER	Specifies the amount of data to be sent to a group member relative to the other members listed for GROUP_MEMBERS when using ROUND_ROBIN_BY_BYTES or ROUND_ROBIN_BY_FILES for ROTATION_STRATEGY. The MULTIPLIER parameter can contain multiple floating-point values: • If the number of MULTIPLIER values equals the number of GROUP_MEMBERS entries, the values will be used in order for each specified member. • If there are fewer MULTIPLIER values than GROUP_MEMBERS entries, the last value will be repeated for the remaining members.

- If there are more values in MULTIPLIER than there are entries in GROUP_MEMBERS, the extras are ignored (and dmcheck will issue a warning).
- If there is no MULTIPLIER parameter, then by default a value of 1 will be used for each MSP/VG in GROUP_MEMBERS. This results in an equal distribution of data among all non-full group members.

Do not change this parameter while DMF is running.

ROTATION_STRATEGY

Specifies the method by which a group member is selected for a migration request. Valid methods are:

- ROUND_ROBIN_BY_BYTES specifies that a certain number of bytes (defined by MULTIPLIER) are sent to each non-full MSP/VG member specified in GROUP_MEMBERS.
- ROUND_ROBIN_BY_FILES specifies that a certain number of files (defined by MULTIPLIER) are sent to each non-full MSP/VG member specified in GROUP_MEMBERS.
- SEQUENTIAL selects the first member in the list that is not already marked as full. This strategy is the default.

If ROTATION_STRATEGY is set to SEQUENTIAL, all GROUP_MEMBERS except the last must be able to report when they are full:

- For a disk MSP, you should specify FULL_THRESHOLD_BYTES to a non-zero value.
- For a VG, you should specify RESERVED_VOLUMES. See the recommendations in "Configure Appropriately for SGI 400 VTL or COPAN MAID Shelves" on page 109.
- Because a cloud MSP, disk cache manager (DCM) MSP, or FTP MSP never reports that it is full, if used

it must be the last member in the GROUP_MEMBER list.

For more information, see:

- "Configure Appropriately for SGI 400 VTL or COPAN MAID Shelves" on page 109
- "volumegroup Object" on page 410
- "msp Object for a Disk MSP" on page 466

Note the following for ROUND_ROBIN_BY_BYTES and ROUND_ROBIN_BY_FILES:

- The goal of these parameters is to optimize VG bandwidth.
- The amounts specified are rounded up to a whole file or byte boundary.
- When an MSP/VG becomes full, its multiplier is removed from the round-robin calculation and the files are spread among the remaining non-full MSPs/VGs. A disk MSP will only report that it is full when FULL_BYTE_THRESHOLD is configured; a VG will only report that it is full when RESERVED_VOLUMES is configured. (Cloud MSPs, FTP MSPs, and DCM MSPs never report that they are full.)
- The statistics for these strategies are stored in the *SPOOL_DIR* directory on a per-MG basis and are persistent in nature.

Do not change this parameter while DMF is running.

Note: VGs only report that they are full when RESERVED_VOLUMES is specified; disk MSPs only report that they are full when FULL_THRESHOLD_BYTES is specified as a non-zero value. Cloud MSPs, DCM MSPs, and FTP MSPs never report that they are full; therefore, if one of these MSPs is to be included as a GROUP_MEMBER in a migrategroup stanza using SEQUENTIAL for ROTATION_STRATEGY, it must be the last member.

---

migrategroup Object Example with Multiple MGs

Example 7-31 migrategroup Object with Multiple MGs

```
define  mg1
        TYPE                    migrategroup
        ROTATION_STRATEGY       ROUND_ROBIN_BY_BYTES
        GROUP_MEMBERS           vg1 vg2
enddef

define  mg2
        TYPE                    migrategroup
        ROTATION_STRATEGY       ROUND_ROBIN_BY_BYTES
        GROUP_MEMBERS           vg3 vg4
enddef
```

Example 7-31 defines two MGs, mg1 and mg2.

There is no MULTIPLIER value, so the default value of 1 will be used.

Single migrategroup Object Example Using the ROUND_ROBIN_BY_BYTES Strategy

Example 7-32 Single migrategroupUsing the ROUND_ROBIN_BY_BYTES Strategy

```
define  mg3
        TYPE                    migrategroup
        ROTATION_STRATEGY       ROUND_ROBIN_BY_BYTES
        GROUP_MEMBERS           vg1 vg2 vg3 vg4
        MULTIPLIER              1  1.5  2  1
enddef
```

In Example 7-32, vg3 is sent twice as much data as vg1 or vg4, and vg2 is sent 1.5 times as much. If vg3 should become full, dmfdaemon will still send 1.5 times more data to vg2 than to vg1 and vg4.

migrategroup Object Example Using the ROUND_ROBIN_BY_FILES Strategy

Example 7-33 migrategroup Using the SEQUENTIAL Strategy

```
define  mg5
        TYPE                    migrategroup
        ROTATION_STRATEGY       SEQUENTIAL
        GROUP_MEMBERS           copan1 copan2 copan3 copan4 lto1
```

```
enddef
```

In the above example, each MSP will be filled before advancing to the next (that is, `copan1` will be filled before advancing to `copan2`). After `copan4` is filled, any subsequent data overflows to the `lto1` library.

`resourcescheduler` Object

This section discusses the following:

- "`resourcescheduler` Object Name" on page 432
- "`resourcescheduler` Object Summary" on page 432
- "`resourcescheduler` Object Parameters" on page 432

`resourcescheduler` Object Name

The name of the `resourcescheduler` object is chosen by the administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

`resourcescheduler` Object Summary

Following is a summary of the potential parameters for the `resourcescheduler` object:

```
resourcescheduler
        TYPE
        ALGORITHM
        PENALTY
        WEIGHT
```

`resourcescheduler` Object Parameters

The entries for a `resourcescheduler` object, one for each DG in a single library, has the following parameters:

Parameter	Description
`TYPE`	Specifies `resourcescheduler` (required name for this type of object). There is no default.

ALGORITHM	Specifies the resource scheduling algorithm to be used, one of: • `fifo` ("first-in, first out") • `weighted_roundrobin` (default)

If you specify `weighted_roundrobin`, the following apply:

Parameter	Description
PENALTY	Modifies the priority of requests from a VG that is not the next one preferred by the round-robin algorithm. It is a multiplier in the range `0.0-1.0`. Low values reduce the priority of the requests from a VG, high values increase the priority of an urgent request from the VG. The default is `0.7` (an urgent request has a little more priority than the preferred request).
WEIGHT	Assigns a weight to one or more VGs. The ratio of these weights to each other (within the one DG) determines the number of opportunities the VG has to obtain drives when they are needed. The weights are integers in the range `1`–`99` and they need not be unique. For efficiency reasons, small numbers are preferred, especially if large numbers of VGs are defined. If a given VG appears on multiple `WEIGHT` lines, the sum of the weights is used as the effective weight for that VG. Any VG that does not appear on a `WEIGHT` line is assigned the default of `5`. If there are no `WEIGHT` lines, all VGs will use this default, resulting in a strict round-robin behavior. `WEIGHT` has the following format: `WEIGHT` *weight vg1 vg2 ...*

`resourcewatcher` Object

This section discusses the following:

- "`resourcewatcher` Object Name" on page 434
- "`resourcewatcher` Object Summary" on page 434
- "`resourcewatcher` Object Parameters" on page 434

resourcewatcher Object Name

The name of the resourcewatcher object is chosen by the administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

resourcewatcher Object Summary

Following is a summary of the potential parameters for the resourcewatcher object:

```
resourcewatcher
        TYPE
        HTML_REFRESH
```

resourcewatcher Object Parameters

The resourcewatcher object is needed only if you wish to change its parameter defaults; a reference to a resource watcher by the libraryserver object is sufficient to activate it.

The resourcewatcher object has the following parameters:

Parameter	Description
TYPE	Specifies resourcewatcher (required name for this type of object). There is no default.
HTML_REFRESH	Specifies the refresh rate (in seconds) of the generated HTML pages. The default is 60.

allocationgroup Object

The allocationgroup object is optional. You should specify it if you want to change the default value of its parameter. This section discusses the following:

- "allocationgroup Object Name" on page 435
- "allocationgroup Object Summary" on page 435
- "allocationgroup Object Parameters" on page 435

allocationgroup Object Name

The name of the allocationgroup object is chosen by the administrator and may contain up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space. It must match the value of the ALLOCATION_GROUP parameter in the volumegroup stanza; see "volumegroup Object Parameters" on page 411.

allocationgroup Object Summary

Following is a summary of the potential parameters for the allocationgroup object:

```
allocationgroup
        TYPE
        VOL_MSG_TIME
```

allocationgroup Object Parameters

The allocationgroup object is needed only if you wish to change its parameter defaults. A reference to an ALLOCATION_GROUP parameter by a volumegroup object is sufficient to activate an AG.

The allocationgroup object has the following parameters:

Parameter	Description
TYPE	Specifies allocationgroup (required name for this type of object). There is no default.
VOL_MSG_TIME	Specifies the minimum interval (in seconds) between operator notifications for low–volume and no-volume conditions for this AG. DMF will send at most one message for each occurrence of a low-volume condition and one message for each occurrence of a no-volume condition. An AG's low-volume threshold depends on the number of unused volumes in the VGs that use it. You can use this parameter to further reduce the number of messages sent. The default value is 86400

seconds (24 hours), the minimum value is `0`, and the maximum value is `2147483647`.

Examples of Configuring an LS

This section contains the following:

- "Example: LS with a Resource Watcher, Two DGs, and an AG" on page 436
- "Example: LS for Fast-Mount Cache" on page 439

Example: LS with a Resource Watcher, Two DGs, and an AG

Example 7-34 defines an LS containing a default resource watcher, two DGs, and one AG that serves multiple VGs.

Note: Example 7-34 does not use all of the possible options for configuring a `libraryserver` object.

Example 7-34 `libraryserver` Object with a Resource Watcher, Two DGs, and an AG

```
define  ls1
                TYPE                    libraryserver
                COMMAND                 dmatls
                DRIVE_GROUPS            dg1 dg2
                CACHE_SPACE             500m
                TASK_GROUPS             ls_tasks
                WATCHER                 rw
enddef

define  dg1
                TYPE                    drivegroup
                VOLUME_GROUPS           vg_1 vg_2
                MOUNT_SERVICE           openvault
                MOUNT_SERVICE_GROUP     ultrium3grp
                OV_INTERCHANGE_MODES    compression
                DRIVE_SCHEDULER         rs
                DRIVES_TO_DOWN          2
                REINSTATE_DRIVE_DELAY   60
enddef
```

```
define  dg2
                TYPE                    drivegroup
                VOLUME_GROUPS           vg_ul4
                MOUNT_SERVICE           openvault
                MOUNT_SERVICE_GROUP     ultrium4grp
                OV_INTERCHANGE_MODES    compression
                DRIVES_TO_DOWN          2
                REINSTATE_DRIVE_DELAY   60
enddef

define  rs
                TYPE                    resourcescheduler
                WEIGHT                  10      vg_1
                WEIGHT                  5       vg_2
enddef

define  vg_1
                TYPE                    volumegroup
                ALLOCATION_GROUP        ag_ult3
enddef

define  vg_2
                TYPE                    volumegroup
                ALLOCATION_GROUP        ag_ult3
                DRIVE_MAXIMUM           2
enddef

define vg_ul4
                TYPE                    volumegroup
enddef
```

In the above example:

- The `define` value must match the value set previously in the `LS_NAMES` or `MSP_NAMES` parameter of the `dmdaemon` object.
- `COMMAND` is set to `dmatls`, as required.
- There are two DGs, `dg1` and `dg2`:
  - `dg1` contains two VGs (`vg_1` `vg_2`) sharing an AG. A resource scheduler is defined to give primary `vg_1` twice the priority of secondary `vg_2` when

competing for drives. The `volumegroup` objects are slightly different, reflecting that `vg_2` is usually write-only.

The `vg_2` object specifies that it can use at most two tape drives, so that other drives in the `dg1` DG will be immediately available for use by `vg_1` when it needs them.

– `dg2` contains a single VG, `vg_ul4`.

For each VG listed for a `VOLUME_GROUPS` parameter of a `drivegroup` object, there must be a corresponding `volumegroup` object.

- The LS can use 500 million bytes of disk cache space when merging chunks from sparse volumes.
- The `ls_tasks` object (defined elsewhere) will specify how periodic maintenance tasks are completed. For more information, see "LS Tasks" on page 441.
- The `rw` resource watcher allows observation of LS operation through a web browser. Assuming that `SPOOL_DIR` was set in the `base` object to be `/dmf/spool`, the URL is `file://dmf/spool/ls/_rw/ls.html`. Text files are generated in the same directory as the HTML files. (You should define a `resourcewatcher` object only if you want to change its default parameters. See "`resourcewatcher` Object Parameters" on page 434.)
- OpenVault is the mounting service. (Because OpenVault is the default mounting service, this line could be omitted.)
- For `dg1`, OpenVault will use the group name `ultrium3grp`; for `dg1`, OpenVault will use the group name `ultrium4grp`.
- Both drives will be used in compression mode.
- `dg1` overrides the default drive scheduler behavior by referring to an object named `rs`. The `rs` object is a `resourcescheduler` object; it specifies that when there are more requests for drives than there are drives in the DG, `vg_1` (with a weight of `10`) is to be given access twice as often as `vg_2` (with a weight of `5`).

Note: The ratio of the numbers is important, but the exact values are not; the values `40` and `20` would have the same affect.

- Each DG can have at most two drives down temporarily for up to 60 minutes; this allows for recovery from I/O errors if the drives are faulty and will result in an

operation that is more reliable. If a drive goes down, the administrator is e-mailed so that maintenance can be performed.

- There is an AG for Ultrium 3 tapes called `ag_ult3` that is used by VGs `vg_1` and `vg_2` (there is no separate configuration stanza for an AG). No AG is defined for Ultrium 4 tapes in VG `vg_ul4`. The volumes have been assigned to `ag_ult3` by using the `dmvoladm`(8) command, as described in "`volumegroup` Object Example with an AG" on page 423.

  In this case, the volumes will automatically be transferred to either `vg_1` or `vg_2` as they are needed and can be immediately returned from the VG to the AG (subject to the restrictions imposed by the configuration parameters `ALLOCATION_MAXIMUM` and `ALLOCATION_MINIMUM`, which in this case are not defined and therefore use their default values of no allocation maximum and an allocation minimum of `0`).

Example: LS for Fast-Mount Cache

Example 7-35 shows various extracts from the configuration file that highlight some of the configuration objects that are specifically associated with the fast-mount cache feature, using two shelves of a COPAN MAID cabinet as the fast-mount cache in conjunction with secondary storage on a physical tape library.

Example 7-35 `libraryserver` and Associated Objects for Fast-Mount Cache

```
define  daemon
        TYPE                    dmdaemon
        MIGRATION_LEVEL         auto
        LS_NAMES                copan_ls tape_ls
        TASK_GROUPS             daemon_tasks dump_tasks fmc_task
        MOVE_FS                 /dmf/move
enddef

define  copan_ls
        TYPE                    libraryserver
        DRIVE_GROUPS            dg_c00 dg_c01
        COMMAND                 dmatls
enddef


define  vg_policy
        TYPE                    policy
```

```
        SELECT_VG               mg_fmc mg0 mg1
enddef


define  copan_fmc
        TYPE                    fastmountcache
        CACHE_MEMBERS           mg_fmc
enddef

define  mg_fmc
        TYPE                    migrategroup
        GROUP_MEMBERS           vg_c00 vg_c01
        ROTATION_STRATEGY       ROUND_ROBIN_BY_BYTES
enddef

define  dg_c00
        TYPE                    drivegroup
        VOLUME_GROUPS           vg_c00
        MOUNT_SERVICE           openvault
        MOUNT_SERVICE_GROUP     dg_c00
enddef

define  vg_c00
        TYPE                    volumegroup
        MERGE_THRESHOLD         0
        RESERVED_VOLUMES        1
 enddef

define  fmc_task
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_fmc_free.sh at 23:00
        FMC_NAME                copan_fmc
        FREE_VOLUME_MINIMUM     10
        FREE_VOLUME_TARGET      20
enddef
```

In the above example:

- There are two LSs:
  - One for the fast-mount cache (`copan_ls`), which must be listed first

  - One for the secondary–storage data copy (`tape_ls`)

- The `dmdaemon` object has a task for the fast-mount cache operations (`fmc_task`).
- There are two DGs (`dg_c00` and `dg_c01`) associated with the LS for the fast-mount cache.
- The MG for the fast-mount cache (`mg_fmc`) is included in the VG selection policy as are two secondary-storage targets (`mg0` and `mg1`).
- There is an object of type `fastmountcache` (named `copan_fmc`) that has the `mg_fmc` MG as the only member.
- The `mg_fmc` MG contains two VGs (`vg_c00` and `vg_c01`).
- The `dg_c00` DG manages pool of drives in the `vg_c00` volume.
- Volumes within the `vg_c00` VG will never be merged because the `MERGE_THRESHOLD` is set to `0`, as required for volumes in a fast-mount cache.
- The `RESERVED_VOLUMES` parameter is set to `1` in `vg_c00` to ensure proper distribution of data, because `vg_c00` is part of the `mg_fmc` MG listed for `CACHE_MEMBERS`.
- Volumes in the fast-mount cache VGs (`vg_c00` and `vg_c01`) will be freed as required by the `run_fmc_free.sh` task at 11:00 PM each day. When fewer than 10% of the volumes in the fast-mount cache are free, DMF will free the volumes with the oldest write dates until 20% of the volumes are free. For more information, see "`taskgroup` Object Example for Fast-Mount Cache Tasks" on page 339.

Note: For brevity, this example does not show the definitions for `vg_c01`, `dg_c01`. `mg0`, `mg1`, and `tape_ls`.

LS Tasks

This section discusses the following:

- "Overview of LS Tasks" on page 442
- "LS `taskgroup` Object with One VG" on page 443
- "LS `taskgroup` Object with Multiple VGs" on page 444

Overview of LS Tasks

You can configure parameters for how the LS daemon performs the following maintenance tasks:

- Merging sparse volumes with the `run_tape_merge.sh` task (for physical tapes, COPAN VTL virtual tapes, COPAN MAID volumes, and JBFS volumes), and the `THRESHOLD`, `VOLUME_LIMIT`, and `DATA_LIMIT` parameters

  Note: For a VG used as a fast-mount cache, do not configure merge tasks. See "Use Fast-Mount Cache Appropriately" on page 112.

- Stopping volume merges at a specified time with the `run_merge_stop.sh` task

Table 7-1 on page 305 provides a summary of automated maintenance tasks. For each of these tasks, you can configure when the task is run. For merging sparse volumes, you must provide more information such as what determines that a volume is sparse and how many volumes can be merged at one time.

Note: The `run_remove_journals.sh` and `run_remove_logs.sh` tasks are configured as part of the `taskgroup` object for daemon tasks, but these tasks also clear the journals and logs for MSPs/LSs. These tasks are described in "`taskgroup` Object" on page 304.

The `run_daily_drive_report.sh`, `run_daily_tsreport.sh`, and `run_daily_report.sh` tasks should be configured as part of the `taskgroup` object for `dmdaemon` tasks. This is because there could be multiple LSs for which `run_daily_drive_report.sh` and `run_daily_tsreport.sh` create reports, and `run_daily_report.sh` reports on other things besides LS information (such as information about the managed filesystems).

LS taskgroup Object with One VG

Note: When modifying sample RUN_TASK parameters, you can comment out any tasks you do not want to run, but you should not change the pathnames or task names, such as $ADMINDIR/run_tape_merge.sh.

Example 7-36 taskgroup Object for LS with One VG

```
define libraryserver_tasks
                TYPE                    taskgroup
                RUN_TASK                $ADMINDIR/run_tape_merge.sh on \
                                        monday wednesday friday at 2:00
                THRESHOLD               50
              # VOLUME_LIMIT             20
              # DATA_LIMIT               5g
                RUN_TASK                $ADMINDIR/run_merge_stop.sh at 5:00
```

In the above example:

- The define value must match the value set previously for the TASK_GROUPS parameter of the libraryserver object. In this case, libraryserver_tasks.
- The run_tape_merge.sh task merges sparse volumes, using the following criteria to determine that a volume is sparse:
  - Its active data is less than 50% (THRESHOLD)
  - There is no limit to the number of volumes that can be selected for merging at one time, because the VOLUME_LIMIT parameter is commented out. (If the comment character is removed, the limit will be 20 volumes.)

    Note: This example uses the run_merge_stop.sh task used to control volume merging rather than the VOLUME_LIMIT and DATA_LIMIT parameters.

  - There is no maximum limit on the amount of data that can be selected for merging at one time because the DATA_LIMIT parameter is commented out. (If the comment character is removed, at most 5 GB can be selected for merging at one time.)
- The run_merge_stop.sh task will shut down volume merging at 5:00 AM every day. Merge requests that were assigned to mover processes will be allowed to complete.

For more information about RUN_TASK parameter, see "taskgroup Object Parameters" on page 312.

LS taskgroup Object with Multiple VGs

For an LS, you can configure volume merging as either of the following:

- As part of the libraryserver object's TASK_GROUPS parameter. This permits volumes from any of the VGs in the LS to be marked as sparse. However, this can lead to drive scheduling and cache usage conflicts.
- As part of a RUN_TASK parameter in the volumegroup object. This avoids the scheduling and conflict problems, but you must ensure that there is no overlap in the times that the various merge tasks run. This might become cumbersome when there are large numbers of VGs configured; in this case, you can use run_merge_mgr.sh rather than run_tape_merge.sh.

The run_merge_mgr.sh task establishes the needs of the VGs for more volumes, using their MIN_VOLUMES parameters as a guide to expected requirements. The task processes the most urgent requests first, minimizing interference with the production workload. To use this task, do the following:

1. Define a taskgroup object, which is referred to by the drivegroup object (not the volumegroup or libraryserver object).
2. Specify a RUN_TASK parameter for run_merge_mgr.sh in the taskgroup object and (optionally) another for run_merge_stop.sh. You can also specify MESSAGE_LEVEL, THRESHOLD, VOLUME_LIMIT, and DATA_LIMIT parameters.
3. Ensure that the libraryserver object that refers to this DG has a resourcewatcher object defined via the WATCHER parameter.
4. For each volumegroup object, confirm that the value of its MIN_VOLUMES parameter is realistic.

LS Database Records

After you have added the LS information to the configuration file, use the dmvoladm(8) command with the -m option to create any missing directories with the proper labels and to create volume (VOL) and catalog (CAT) records in the LS database.

You can follow the steps in Procedure 7-1 for each LS that you have defined.

Caution: Each LS must have a unique set of VSNs.

Procedure 7-1 Creating LS Database Records

The following procedure is shown as an example that assumes you have an LS called `ls1`. This LS contains a VG named `vg_pri`.

1. Enter the following command and it will respond as shown:

   ```
   % dmvoladm -m ls1
   dmvoladm: at rdm_open - created database libsrv_db
   adm: 1>
   ```

   The response is an informational message indicating that `dmvoladm` could not open an existing LS database, so it is creating a new and empty one. You should get this message the first time you use `dmvoladm` for an LS, but never again. The next line (`adm:1>`) is the prompt for `dmvoladm` directives.

2. Assume that you will use 200 volumes with VSNs `VA0001` through `VA0200`. After the prompt, enter the following directive:

   ```
   adm:1> create VA0001-VA0200 vg vg_pri
   ```

   Note: You are specifying the VG `vg_pri` for the volumes being added. It is also valid to specify an AG name instead of a VG name.

   After entering this directive, you will receive 200 messages, one for each entry created, beginning with the following:

   ```
   VSN VA0001 created.
   VSN VA0002 created.
   ```

3. List all of the VSNs in the newly created library:

   ```
   adm:2> list all
   ```

4. Complete setting up the LS:

   ```
   adm:3> quit
   ```

msp Objects

This section discusses the following:

- "msp Object Name" on page 446
- "NAME_FORMAT Details" on page 446
- "msp Object for a Cloud MSP" on page 448
- "msp Object for an FTP MSP" on page 455
- "msp Object for a DCM MSP" on page 459
- "msp Object for a Disk MSP" on page 466

msp Object Name

The name of an msp object (for a cloud, FTP, disk, or DCM MSP) is chosen by the administrator and may contain and up to 8 uppercase or lowercase alphanumeric characters or underscores. It cannot begin with an underscore or contain any white space.

NAME_FORMAT Details

The NAME_FORMAT parameter consists of substitution strings to create a template for the names that are used in secondary storage. It applies to the following:

- Cloud MSP for S3 systems
- FTP MSP
- disk MSP
- DCM MSP

The default NAME_FORMAT for all of the supported MSPs is %u/%b (*username/bfid*). However, this default may not work well in all circumstances, particularly for a large number of files in a given directory. You can use the substitution strings listed in Table 7-3 to create a template that is more appropriate for your site.

The NAME_FORMAT value must include one of the following:

- %b (which will guarantee a unique filename)

- %2, %3, %4 in some combination

The default size allotted to the NAME_FORMAT value in the daemon database record is 34 bytes. This is large enough to accommodate the default for NAME_FORMAT if the user name is 8 or fewer characters (the %b value is always 24 characters). If you choose a set of strings that will evaluate to a field that is larger than 34 bytes, you may want to consider increasing the size of this record; see "Daemon Database Record Length" on page 173.

The NAME_FORMAT value is only used when files are being stored. After a file has been stored, the name is saved in the daemon database. This saved name is used to retrieve the file. This ensures that migrated files remain accessible even if they are renamed, if their owner changes, or if the NAME_FORMAT value changes. If you change the value, it affects only subsequent store operations.

Note the following considerations for specific MSPs:

- For the S3 cloud MSP, you can use the %p string to randomize the order of keys, which may improve performance for a bucket with a high ingest rate. For example, a NAME_FORMAT value of %p%p/%b will create keys that are randomly distributed according to their first two characters. For more information, see:

  http://aws.amazon.com/articles/Amazon-S3/1904

- For a disk MSP, DCM MSP, or FTP MSP, you can use the %p or the date- and time-related substitution strings to spread files over a large number of directories, to potentially minimize subsequent access times.
- For an FTP MSP that is implemented on a system other than one based on UNIX or Linux, the default may not work.

Table 7-3 NAME_FORMAT Strings

String Type	String	Evaluates To
BFID strings	%1	First set of 32 bits of the BFID in lowercase hexadecimal, which are always 8 pad characters (00000000)
	%2	Second set of 32 bits of the BFID in lowercase hexadecimal

String Type	String	Evaluates To
	%3	Third set of 32 bits of the BFID in lowercase hexadecimal
	%4	Fourth set of 32 bits of the BFID in lowercase hexadecimal
	%b	BFID in lowercase hexadecimal (least-significant 24 characters) without the 8 pad characters found in the 8 most-significant characters of the full BFID
Ownership strings	%u	User name of the file owner
	%U	User ID of the file owner
	%g	Group name of the file
	%G	Group ID of the file
Date strings	%d	Current day of month (2 digits)
	%H	Current hour (2 digits)
	%m	Current month (2 digits)
	%M	Current minute (2 digits)
	%S	Current second (2 digits)
	%y	Last 2 digits of the current year (such as 03 for 2003)
Percent string	%%	Literal % character
Partition string	%p	A digit (0 through 9) derived from the BFID, producing an essentially random number; up to 9 %p strings are permitted

msp Object for a Cloud MSP

This section discusses the following:

- "msp Object Summary for a Cloud MSP" on page 449
- "msp Object Parameters for a Cloud MSP" on page 449
- "msp Object Example for an S3 Cloud MSP" on page 453
- "msp Object Example for a Scality Cloud MSP" on page 454

msp Object Summary for a Cloud MSP

Following is a summary of the potential parameters for the msp object for a cloud MSP:

```
msp
        TYPE
        CHILD_MAXIMUM
        CLOUD_BACKEND
        CLOUD_BASE_PATH
        CLOUD_BUCKET
        CLOUD_HOST
        CLOUD_PASSWORD
        CLOUD_REDUNDANCY
        CLOUD_USER
        COMMAND
        GUARANTEED_DELETES
        GUARANTEED_GETS
        MESSAGE_LEVEL
        NAME_FORMAT
        TASK_GROUPS
        WRITE_CHECKSUM
```

msp Object Parameters for a Cloud MSP

To enable a cloud MSP, define an msp object for it in the DMF configuration file and include the name of that object on the MSP_NAMES or LS_NAMES parameter in the dmdaemon object.

The MSP checks the DMF configuration file just before it starts child processes. If the DMF configuration file changed, it is reread.

A cloud msp object has the following parameters:

Parameter	Description
TYPE	Specifies msp (required name for this type of object). There is no default.
CHILD_MAXIMUM	Specifies the maximum number of child processes that the MSP is allowed to fork. The legal range of values is 0–100; the default is 4.

If CHILD_MAXIMUM is nonzero, its value must be greater than the sum of GUARANTEED_DELETES and GUARANTEED_GETS.

CLOUD_BACKEND

Specifies the backend used by this cloud instance, which determines how other cloud-specific parameters are interpreted. There is no default value. Legal values are:

s3	Specifies Amazon Simple Storage Service (S3)
scality	Specifies Scality RING

CLOUD_BASE_PATH

(S3 only) Specifies a string that is prefixed to the key (separated from the key by a slash "/" character) in all requests to S3. The string may not begin or end with a slash (/). By default, there is no prefix.

A prefix may be useful if another application is sharing the same S3 bucket and you want to differentiate the keys that belong to data migrated by DMF. For example, if this parameter is set to DMF, the S3 key name would begin with DMF/*NAME_FORMAT*.

CLOUD_BUCKET

(S3 only) Specifies the name of the existing S3 bucket to use when storing files. There is no default for S3 systems.

CLOUD_HOST

Specifies one or more domain names or IP addresses used to connect to the cloud, using the following format:

hostname1[:*port*][,*hostname2*[:*port*]]

You can specify multiple comma-separated addresses. Connections will be distributed in round-robin fashion, in an order that is randomized from the order specified. The cloud MSP will attempt to send requests to all of the addresses, ignoring any that fail to respond.

You must provide at least one *host* value:

- S3: *host* specifies the address of the S3 service.
- Scality: *host* specifies the machines that are running the Scality sproxyd daemons

The default for *port* varies by the cloud service:

- S3: `80`
- Scality: `81`

`CLOUD_PASSWORD` — Specifies the file containing password information to use on the cloud:

- S3: specifies the location of the file that contains the Secret Access Key (required for S3)
- Scality: specifies the file containing the password to use on the cloud host (required only if HTTP Basic Authentication is enabled on the cloud host; see the Scality documentation for details)

If this file exists, it must be owned by `root` and only be accessible by `root`.

`CLOUD_REDUNDANCY` — Specifies the level of data redundancy that is requested from the cloud service for new files.

Valid values for S3 are as follows:

S3 Value	Description
`reduced`	Specifies S3 *Reduced Redundancy Storage* (RRS)
`standard`	Specifies standard S3 redundancy

The default for S3 is `standard`.

Valid values for Scality are as follows:

Scality Value	Description
`1copy`	Specifies that only a single copy of the data is kept (no redundancy)
`2copies` to `6copies`	Specifies the number of copies (from two to six) that will be kept

ARC	Specifies Scality *Advanced Resilience Configuration* (ARC)

The default for Scality is `3copies`.

Note: SGI strongly recommends that you migrate at least two copies to secondary-storage targets in order to prevent file data loss in the event that a migrated copy is damaged. A given cloud can be a single point of failure, therefore redundant copies within one cloud do not sufficiently protect against data loss. SGI therefore highly recommends that you migrate data to a second location (to another cloud instance, tape, or disk).

`CLOUD_USER` Specifies the username information on the cloud host:

- S3: specifies the Access Key ID (required)
- Scality: specifies the cloud user (required only if HTTP Basic Authentication is enabled on the cloud host; see the Scality documentation for details)

`COMMAND` Specifies the binary file to execute in order to initiate this MSP. For the cloud MSP, this value must be `dmcloudmsp`. There is no default.

`GUARANTEED_DELETES` Specifies the number of child processes that are guaranteed to be available for processing delete requests. If `CHILD_MAXIMUM` is nonzero, its value must be greater than the sum of `GUARANTEED_DELETES` and `GUARANTEED_GETS`. The default is `1`.

`GUARANTEED_GETS` Specifies the number of child processes that are guaranteed to be available for processing `dmget`(1) requests. If `CHILD_MAXIMUM` is nonzero, its value must be greater than the sum of `GUARANTEED_DELETES` and `GUARANTEED_GETS`. The default is `1`.

`MESSAGE_LEVEL` Specifies the highest message level that will be written to the MSP log. It must be an integer in the range `0`–`6`; the higher the number, the more messages written to the log file. The default is `2`. For more information on

	message levels, see Chapter 10, "Message Log Files" on page 507.
NAME_FORMAT	*(S3 only)* Specifies substitution strings that provide a template for keys on the S3 cloud storage. You can customize the strings to create a template that is appropriate for your site; see "NAME_FORMAT Details" on page 446.
TASK_GROUPS	Names the taskgroup objects that contain tasks the MSP should run. By default, no tasks are run.
WRITE_CHECKSUM	Specifies if the cloud MSP's copy of the file should be checksummed before writing. If the file has been checksummed, it is verified when read. You can set this parameter to ON or OFF. The default is ON.

msp Object Example for an S3 Cloud MSP

Example 7-37 msp Object for an S3 Cloud MSP

```
define cloudA
        TYPE                    msp
        COMMAND                 dmcloudmsp
        CLOUD_BACKEND           s3
        CLOUD_HOST              s3.amazonaws.com
        CLOUD_USER              WITST4TQBFJOTLDB2IAB
        CLOUD_PASSWORD          HOME_DIR/password
        CLOUD_BUCKET            dmf-bucket
        NAME_FORMAT             %p%p/%b
enddef
```

In the above example:

- The define value must match the MSP_NAMES or LS_NAMES parameter of the dmdaemon object.
- The command to initiate the cloud MSP must be dmcloudmsp.
- This cloud MSP instance uses S3.
- The name of the cloud host on which files will be stored is s3.amazonaws.com and the default port will be used.

- The Access Key ID for the cloud user during session initialization is `WITST4TQBFJOTLDB2IAB`.
- The Secret Access Key is stored in the file *HOME_DIR*`/password`.
- The S3 bucket is named `dmf-bucket`.
- The key names will be randomly ordered by the first two characters (which will be anywhere from `00` through `99`). This avoids sequential keys when writing files, which may potentially minimize subsequent access times and improve performance for a bucket with a high ingest rate.
- Because `CLOUD_REDUNDANCY` is not specified, the default `standard` redundancy will be used.

`msp` Object Example for a Scality Cloud MSP

Example 7-38 `msp` Object for a Scality Cloud MSP

```
define cloudB
        TYPE                    msp
        COMMAND                 dmcloudmsp
        CLOUD_BACKEND           scality
        CLOUD_HOST              scalityserver
        CLOUD_REDUNDANCY        3copies
enddef
```

In the above example:

- The `define` value must match the `MSP_NAMES` or `LS_NAMES` parameter of the `dmdaemon` object
- The command to initiate the cloud MSP must be `dmcloudmsp`
- This cloud MSP instance uses Scality
- The name of the cloud host on which files will be stored is `scalityserver`
- Three copies will be made by the cloud service for redundancy

msp Object for an FTP MSP

This section discusses the following:

- "msp Object Summary for an FTP MSP" on page 455
- "msp Object Parameters for an FTP MSP" on page 455
- " msp Object Example for an FTP MSP" on page 458

msp Object Summary for an FTP MSP

Following is a summary of the potential parameters for the msp object for an FTP MSP:

```
msp
        TYPE
        CHILD_MAXIMUM
        COMMAND
        FTP_ACCOUNT
        FTP_COMMAND
        FTP_DIRECTORY
        FTP_HOST
        FTP_PASSWORD
        FTP_PORT
        FTP_USER
        GUARANTEED_DELETES
        GUARANTEED_GETS
        IMPORT_DELETE
        IMPORT_ONLY
        MESSAGE_LEVEL
        MVS_UNIT
        NAME_FORMAT
        TASK_GROUPS
        WRITE_CHECKSUM
```

msp Object Parameters for an FTP MSP

To enable a File Transfer Protocol (FTP) MSP, define an msp object for it in the DMF configuration file and include the name of that object on the MSP_NAMES or LS_NAMES parameter in the dmdaemon object.

DMF has the capability to use an FTP MSP to convert a non-DMF fileserver to DMF with a minimal amount of down time for the switch over, and at a site-determined pace. Contact your customer service representative for information about technical assistance with fileserver conversion.

The MSP checks the DMF configuration file just before it starts child processes. If the DMF configuration file changed, it is reread.

An FTP `msp` object has the following parameters:

Parameter	Description
`TYPE`	Specifies `msp` (required name for this type of object). There is no default.
`CHILD_MAXIMUM`	Specifies the maximum number of child processes that the MSP is allowed to fork. The legal range of values is `0`–`100`; the default is `4`. If `CHILD_MAXIMUM` is nonzero, its value must be greater than the sum of `GUARANTEED_DELETES` and `GUARANTEED_GETS`.
`COMMAND`	Specifies the binary file to execute in order to initiate this MSP. For the FTP MSP, this value must be `dmftpmsp`. There is no default.
`FTP_ACCOUNT`	Specifies the account ID to use on the remote FTP server. Most FTP servers do not need account information. By default, no account information is supplied. When account information is required, its nature and format will be dictated by the remote host and will vary from operating system to operating system.
`FTP_COMMAND`	Specifies an additional command to send to the remote system. There may be more than one instance of this parameter. By default, no other commands are sent.
`FTP_DIRECTORY`	Specifies the directory into which files will be placed on the remote FTP server. There is no default.
`FTP_HOST`	Specifies the domain name or IP address of the remote node on which files are to be stored. If you use a domain name with multiple IP addresses, the FTP MSP tries all of the addresses in order. If the remote system

	cannot be reached, the MSP waits 5 minutes and retries again until it succeeds. There is no default.
FTP_PASSWORD	Specifies the file containing the password to use when migrating files to the remote system. This file must be owned by root and be only accessible by root. (The MSP will not operate if the FTP_PASSWORD file is readable by anyone other than root.) There is no default.
FTP_PORT	Specifies the port number of the FTP server on the remote system. The default is the value configured for ftp in the services file.
FTP_USER	Specifies the user name to use when migrating files to the remote system. There is no default.
GUARANTEED_DELETES	Specifies the number of child processes that are guaranteed to be available for processing delete requests. If CHILD_MAXIMUM is nonzero, its value must be greater than the sum of GUARANTEED_DELETES and GUARANTEED_GETS. The default is 1.
GUARANTEED_GETS	Specifies the number of child processes that are guaranteed to be available for processing dmget(1) requests. If CHILD_MAXIMUM is nonzero, its value must be greater than the sum of GUARANTEED_DELETES and GUARANTEED_GETS. The default is 1.
IMPORT_DELETE	Specifies if the MSP should honor hard-delete requests from the daemon. (This parameter applies only if IMPORT_ONLY is set to ON.) You can set this parameter to ON or OFF. The default is OFF. Set IMPORT_DELETE to ON if you wish files to be deleted on the destination system when hard deletes are processed.
IMPORT_ONLY	Specifies if the MSP is used for importing only. You can set this parameter to ON or OFF. The default is OFF. Set this parameter ON when the data is stored as a bit-for-bit copy of the file and must be available to DMF as part of a conversion. The daemon will not accept dmput(1), dmmove(8), or dmarchive(1) requests that specify an MSP with this parameter enabled. By default, the MSP will ignore hard-delete requests when this parameter is enabled. When the DMF daemon performs a complete

file recall from an import-only MSP and all other DMF copies also reside in import-only MSPs or VGs, it makes the file a regular file rather than a dual-state file, and it soft-deletes the MSP's copy of the file.

Note: An import-only MSP should never be a member of a `migrategroup` stanza.

`MESSAGE_LEVEL` — Specifies the highest message level that will be written to the MSP log. It must be an integer in the range `0`–`6`; the higher the number, the more messages written to the log file. The default is `2`. For more information on message levels, see Chapter 10, "Message Log Files" on page 507.

`MVS_UNIT` — Defines the storage device type on an IBM MVS™ system. You must specify a this parameter when the destination is an MVS system. Valid values are:

```
3330
3350
3380
3390
```

`NAME_FORMAT` — Specifies the strings that form a template for names stored in *FTP_DIRECTORY*. You can customize the strings to create a template that is appropriate for your site; see "`NAME_FORMAT` Details" on page 446.

`TASK_GROUPS` — Names the `taskgroup` objects that contain tasks the MSP should run. By default, no tasks are run.

`WRITE_CHECKSUM` — Specifies if the FTP MSP's copy of the file should be checksummed before writing. If the file has been checksummed, it is verified when read. You can set this parameter to `ON` or `OFF`. The default is `ON`.

`msp` Object Example for an FTP MSP

Example 7-39 `msp` Object for an FTP MSP

```
define  ftp
        TYPE                    msp
```

```
        COMMAND                 dmftpmsp
        FTP_HOST                fileserver
        FTP_USER                dmf
        FTP_ACCOUNT             dmf.disk
        FTP_PASSWORD            /dmf/ftp/password
        FTP_DIRECTORY           ftpmsp
        FTP_COMMAND             umask 022
enddef
```

In the above example:

- The string `%u/%b` will be used as a template to create names for files stored on remote hosts in *FTP_DIRECTORY* (which is the default when `NAME_FORMAT` is not specified)
- The `define` value must match the `MSP_NAMES` or `LS_NAMES` parameter of the `dmdaemon` object
- The command to initiate the FTP MSP must be `dmftpmsp`
- The user name for the remote FTP server during session initialization is `dmf`
- The name of the remote host on which files will be stored is `fileserver`
- The remote host requires the FTP account information `dmf.disk`
- The password for the user on the remote host is stored in the file `/dmf/ftp/password`
- Files will be placed into the `ftpmsp` directory on the remote host
- The `umask` for files created will be set to `022`, which removes write permission for `group` and `other`

`msp` Object for a DCM MSP

This section discusses the following:

- "`msp` Object Summary for a DCM MSP" on page 460
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Example for a DCM MSP" on page 465

msp Object Summary for a DCM MSP

Following is a summary of the potential parameters for the msp object for a DCM MSP:

```
msp
        TYPE
        BUFFERED_IO_SIZE
        CHILD_MAXIMUM
        COMMAND
        DIRECT_IO_SIZE
        DSK_BUFSIZE
        DUMP_FLUSH_DCM_FIRST
        FADV_SIZE_MSP
        GUARANTEED_DELETES
        GUARANTEED_GETS
        MESSAGE_LEVEL
        MIGRATION_LEVEL
        MIN_DIRECT_SIZE
        NAME_FORMAT
        POLICIES
        PRIORITY_PERIOD
        STORE_DIRECTORY
        TASK_GROUPS
        WRITE_CHECKSUM
```

msp Object Parameters for a DCM MSP

A DCM MSP is a disk MSP that is configured for *n*–tier capability. To enable a DCM MSP, define an msp object for it in the DMF configuration file and include a name for that object on the MSP_NAMES or LS_NAMES parameter in the dmdaemon object.

Note: The parameters differ for a disk MSP that is not a DCM MSP. See "msp Object for a Disk MSP" on page 466.

A DCM msp object has the following parameters:

Parameter	Description
TYPE	Specifies msp (required name for this type of object). There is no default.

BUFFERED_IO_SIZE	Specifies the size of I/O requests for buffered I/O when migrating files downward in the hierarchy from *STORE_DIRECTORY* of this DCM MSP. The legal range of values is 4096–16777216. The default is 262144. By default, the unit of measure is bytes; see "Units of Measure" on page 275.
CHILD_MAXIMUM	Specifies the maximum number of child processes that the DCM MSP is allowed to fork. The legal range of values is 0–100. The default is 4. **Note:** SGI recommends that you use a larger value than the default for a DCM MSP.
COMMAND	Specifies the binary file to execute in order to initiate this MSP. For the DCM MSP, this value must be dmdskmsp.
DIRECT_IO_SIZE	Specifies the size of I/O requests for direct I/O when migrating files downward in the hierarchy from *STORE_DIRECTORY* for this DCM MSP. The legal range of values is 65536–18446744073709551615. The default depends on the filesystem, but will not exceed the value of DIRECT_IO_MAXIMUM_SIZE defined in the base object (see "base Object" on page 277). By default, the unit of measure is bytes; see "Units of Measure" on page 275. For more information about direct I/O, see O_DIRECT in the open(2) man page.
DSK_BUFSIZE	Specifies the transfer size in bytes used when reading from and writing to files within the DCM MSP *STORE_DIRECTORY*. The value must be in the range 4096–16000000 (16 million). The default is 131072 when writing and 1000000 when reading. By default, the unit of measure is bytes; see "Units of Measure" on page 275.
DUMP_FLUSH_DCM_FIRST	*(integrated-backup context only)* Specifies whether or not the dmmigrate command is run on the filesystem specified by the STORE_DIRECTORY parameter for this

DCM MSP before dumping the filesystem, overriding the value set for the `taskgroup` object. Running `dmmigrate` first ensures that all non-dual-resident files in the DCM MSP caches are migrated to a lower tier. You can set this parameter to `ON` or `OFF`. The default is `ON`. For integrated backups, the value set in the `taskgroup` object overrides the value set here.

`FADV_SIZE_MSP` — Specifies the size of files in the MSP's *STORE_DIRECTORY* for which `posix_fadvise()` will be called with advice `POSIX_FADV_DONTNEED`. If the file is larger than *FADV_SIZE_MSP* bytes, the call is made following migration to the MSP or following recall of the entire file. The minimum is `0`, which means that `posix_fadvise()` will always be called, and the maximum is `9223372036854775807`. The default is `10000000`. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

`GUARANTEED_DELETES` — Specifies the number of child processes that are guaranteed to be available for processing delete requests. The default is `1`.

`GUARANTEED_GETS` — Specifies the number of child processes that are guaranteed to be available for processing `dmget`(1) requests. The default is `1`.

`MESSAGE_LEVEL` — Specifies the highest message level that will be written to the MSP log. It must be an integer in the range `0`–`6`; the higher the number, the more messages written to the log file. The default is `2`. For more information on message levels, see Chapter 10, "Message Log Files" on page 507.

`MIGRATION_LEVEL` — Specifies the level of migration service for the DCM MSP. Valid values are:

- `auto` (automated space management)
- `none` (no flushing to a lower VG)
- `user` (only requests from `dmmigrate` or a manually invoked `dmdskfree`)

The default is `auto`.

MIN_DIRECT_SIZE

Determines whether direct or buffered I/O is used when migrating files downward in the hierarchy from the *STORE_DIRECTORY* of this DCM MSP. If the number of bytes to be read is smaller than the value specified, buffered I/O is used, otherwise direct I/O is used. The legal range of values is `0` (direct I/O is always used) through `18446744073709551615` (direct I/O is never used). The default is `0`. By default, the unit of measure is bytes; see "Units of Measure" on page 275.

Note: For real-time filesystems, this parameter is ignored.

For more information about direct I/O, see `O_DIRECT` in the `open`(2) man page.

NAME_FORMAT

Specifies the substitution strings that form a template for names stored in *STORE_DIRECTORY*. You can customize the strings to create a template that is appropriate for your site; see "`NAME_FORMAT` Details" on page 446.

POLICIES

Specifies the names of the configuration objects defining policies for this filesystem. The configuration stanza must contain at least one `POLICIES` parameter and the configuration stanza for that parameter must contain a `SELECT_LOWER_VG` parameter.

PRIORITY_PERIOD

Specifies the number of minutes after which a migrating file gets special treatment.

Normally, if there is insufficient room in the *STORE_DIRECTORY* for a file, the DCM MSP will attempt to make room, while continuing to store files that will fit. If a file has not been stored into the *STORE_DIRECTORY* within *PRIORITY_PERIOD*, however, the DCM MSP will stop trying to store other files until either sufficient room has been made or it has determined that room cannot be made. The legal range of values is `1`–`2000000`; the default is `120` minutes (2 hours).

STORE_DIRECTORY	Specifies the directory used to hold files for this DCM MSP. This directory must not be in a managed filesystem. The directory must be the mount point of a dedicated XFS or CXFS filesystem mounted with DMAPI enabled. You must mount a DCM MSP STORE_DIRECTORY with both dirsync and dmi. In addition, when using Parallel DMF, the directory must be a CXFS filesystem. See "Configure Filesystems and Directories Appropriately for DMF" on page 94.
TASK_GROUPS	Names the taskgroup objects that contain tasks the DCM MSP should run. By default, no tasks are run.
WRITE_CHECKSUM	Specifies if the DCM MSP's copy of the file should be checksummed before writing. If the file has been checksummed, it is verified when read. You can set this parameter to ON or OFF. The default is ON.

A DCM MSP also requires a task group that runs the run_dcm_admin.sh task during off-peak hours to perform routine maintenance for the DCM MSP.

When using a DCM MSP, dmdskmsp will not fail if the *STORE_DIRECTORY* is full. Instead, it will queue the requests and wait to fulfill them until after dmdskfree has freed the required space.

msp Object Example for a DCM MSP

Following is a sample of the configuration stanzas with some explanatory notes below. Many of parameters have defaults and can be omitted if the defaults are appropriate.

Example 7-40 Configuration Stanzas Associated with a DCM MSP

```
define  daemon
        TYPE                    dmdaemon
        LS_NAMES                dcm_msp ls                      ### See note 1
        ...                                                     ### See note 2
enddef

define  msp_policy
        TYPE                    policy
        SELECT_MSP              dcm_msp copy2 when space > 4096 ### See note 3
        ...                                                     ### See note 2
enddef

define  dcm_msp
        TYPE                    msp
        COMMAND                 dmdskmsp
        STORE_DIRECTORY         /dmf/dcm_msp_store              ### See note 4
        CHILD_MAXIMUM           10                              ### See note 5
        POLICIES                dcm_policy
        TASK_GROUPS             dcm_tasks
enddef

define  dcm_policy
        TYPE                    policy                          ### See note 6
        FREE_SPACE_MINIMUM      10
        FREE_SPACE_TARGET       70
        DUALRESIDENCE_TARGET    90
        FREE_SPACE_DECREMENT    1
        FREE_DUALRESIDENT_FIRST on
        CACHE_AGE_WEIGHT        1       .1
        CACHE_SPACE_WEIGHT      1       .1
        SELECT_LOWER_VG         none when uid = 0
        SELECT_LOWER_VG         vg1 when space > 1G
        SELECT_LOWER_VG         vg2
enddef
```

```
define  dcm_tasks
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_dcm_admin.sh at 22:00:10
enddef
```

Notes referred to in the preceding example:

1. The DCM MSP must be specified before the LSs that contain its lower VGs. (Otherwise, all recalls will attempt to come directly from the lower VGs.)
2. Other parameters essential to the use of this stanza but not relevant to the DCM MSP have been omitted.
3. The DCM MSP and its lower VGs should be considered to act as a single high-speed VG logically maintaining only one copy of a migrated file. You should always have a second copy of all migrated files, which is the purpose of `copy2` in this example. It would probably be a tape VG, but could be any type of MSP other than a DCM MSP.

   The copy that resides in the DCM MSP *STORE_DIRECTORY* is not to be considered a permanent copy of the file in terms of the safety of the file's data. It can be deleted at any time, though never before a copy of it exists in one of the `SELECT_LOWER_VG` VGs.
4. The mount point of a **dedicated** DMAPI-mounted filesystem.
5. Any other parameters applicable to a disk MSP may also be used, with the exception of `IMPORT_ONLY` and `IMPORT_DELETE`.
6. Several parameters in DCM MSP policies have functions that are analogous to those in disk MSP policies; see "Rules for `policy` Parameters" on page 366 and "Managed Filesystem `policy` Parameters" on page 368.

`msp` Object for a Disk MSP

This section discusses the following:

- "`msp` Object Summary for a Disk MSP" on page 467
- "`msp` Object Parameters for a Disk MSP" on page 467
- "`msp` Object Example for a Disk MSP" on page 470

msp Object Summary for a Disk MSP

Following is a summary of the potential parameters for the msp object for a disk MSP:

```
msp
        TYPE
        CHILD_MAXIMUM
        COMMAND
        DSK_BUFSIZE
        FADV_SIZE_MSP
        FULL_THRESHOLD_BYTES
        GUARANTEED_DELETES
        GUARANTEED_GETS
        IMPORT_DELETE
        IMPORT_ONLY
        MESSAGE_LEVEL
        NAME_FORMAT
        STORE_DIRECTORY
        TASK_GROUPS
        WRITE_CHECKSUM
```

msp Object Parameters for a Disk MSP

Note: The parameters differ for a DCM MSP, which is a disk MSP configured for *n*–tier capability. See "msp Object for a DCM MSP" on page 459.

To enable a disk MSP, define an msp object for it in the DMF configuration file and include a name for that object on the MSP_NAMES or LS_NAMES parameter in the dmdaemon object.

You can use a disk MSP to convert a non-DMF fileserver to DMF with a minimal amount of down time for the switch over, and at a site-determined pace. Contact your customer service representative for information about technical assistance with fileserver conversion.

A disk msp object has the following parameters:

Parameter	Description
TYPE	Specifies msp (required name for this type of object). There is no default.

CHILD_MAXIMUM	Specifies the maximum number of child processes that the MSP is allowed to fork. The legal range of values is 0–100. The default is 4.
COMMAND	Specifies the binary file to execute in order to initiate this MSP. For the disk MSP, this value must be dmdskmsp.
DSK_BUFSIZE	Specifies the transfer size in bytes used when reading from and writing to files within the disk MSP's *STORE_DIRECTORY*. The value must be in the range 4096–16000000 (or 16m). The default is 131072 when writing and 1000000 when reading. By default, the unit of measure is bytes; see "Units of Measure" on page 275.
FADV_SIZE_MSP	Specifies the size of files in the MSP's *STORE_DIRECTORY* for which posix_fadvise() will be called with advice POSIX_FADV_DONTNEED. If the file is larger than *FADV_SIZE_MSP* bytes, the call is made following migration to the MSP or following recall of the entire file. The minimum is 0, which means that posix_fadvise() will always be called, and the maximum is 9223372036854775807. The default is 10000000. By default, the unit of measure is bytes; see "Units of Measure" on page 275.
FULL_THRESHOLD_BYTES	Specifies the number of bytes at which point the disk MSP will tell the DMF daemon that it is full. If 0, the disk MSP will never report that it is full. If non-zero, the MSP will report when it is full and will continue to report full until the number of bytes specified by FULL_THRESHOLD_BYTES become free. (You can free bytes by using dmmove(8) or hard-deleting BFIDs.) The SEQUENTIAL choice for ROTATION_STRATEGY relies on the disk MSP reporting that it is full (see "migrategroup Object" on page 426); if the MSP is part of an MG, setting FULL_THRESHOLD_BYTES to a non-zero value prevents the MG from sending migrations to the MSP before enough disk space has been freed to make the migrations productive. If you specify a non-zero value, you should also set

	EXPORT_METRICS to ON (see "base Object" on page 277). The default is 0.
GUARANTEED_DELETES	Specifies the number of child processes that are guaranteed to be available for processing delete requests. The default is 1.
GUARANTEED_GETS	Specifies the number of child processes that are guaranteed to be available for processing dmget(1) requests. The default is 1.
IMPORT_DELETE	Specifies if the MSP should honor hard-delete requests from the daemon. (This parameter only applies if IMPORT_ONLY is set to ON.) You can set this parameter to ON or OFF. The default is OFF. Set IMPORT_DELETE to ON if you want files to be deleted in *STORE_DIRECTORY* when hard deletes are processed.
IMPORT_ONLY	Specifies if the MSP is used for importing only. You can set this parameter to ON or OFF. The default is OFF. Set this parameter ON when the data is stored as a bit-for-bit copy of the file and must be available to DMF as part of a conversion. The daemon will not accept dmput(1), dmmove(8), or dmarchive(1) requests that specify an MSP with this parameter enabled. The MSP will, by default, ignore hard-delete requests when this parameter is enabled. When the DMF daemon performs a complete file recall from an import-only MSP and all other DMF copies also reside in import-only MSPs or VGs, it makes the file a regular file (rather than a dual-state file) and it soft-deletes the MSP's copy of the file. **Note:** An import-only MSP should never be a member of a migrategroup stanza.
MESSAGE_LEVEL	Specifies the highest message level that will be written to the MSP log. It must be an integer in the range 0–6; the higher the number, the more messages written to the log file. The default is 2. For more information on message levels, see Chapter 10, "Message Log Files" on page 507.

NAME_FORMAT	Specifies the substitution strings that form a template for names stored in *STORE_DIRECTORY*. You can customize the strings to create a template that is appropriate for your site; see "NAME_FORMAT Details" on page 446.
STORE_DIRECTORY	Specifies the directory that holds files for this disk MSP. This directory may be in a local filesystem or in a remote filesystem that has been imported by NFS, CXFS, or similar. This directory must not be in a managed filesystem. By default, this value must not reside in the root filesystem (see the description of ADMDIR_IN_ROOTFS in "base Object Parameters" on page 278). You must mount a disk MSP STORE_DIRECTORY with dirsync (but without dmi); see "Configure Filesystems and Directories Appropriately for DMF" on page 94.
TASK_GROUPS	Names the taskgroup objects that contain tasks the MSP should run. By default, no tasks are run.
WRITE_CHECKSUM	Specifies if the disk MSP's copy of the file should be checksummed before writing. If the file has been checksummed, it is verified when read. You can set this parameter to ON or OFF. The default is ON.

msp Object Example for a Disk MSP

Example 7-41 msp Object for a Disk MSP

```
define  dsk
        TYPE                    msp
        COMMAND                 dmdskmsp
        CHILD_MAXIMUM           8
        GUARANTEED_DELETES      3
        GUARANTEED_GETS         3
        STORE_DIRECTORY         /dmf/dsk_store
enddef
```

In the above example:

- The define value must match the MSP_NAMES or LS_NAMES parameter of the dmdaemon object.

- The command to initiate the disk MSP must be `dmdskmsp`.
- This MSP can fork up to 8 child processes.
- 3 child processes are guaranteed to be available for processing delete and get requests.
- Files will be stored in `/dmf/dsk_store`.

Summary of the Configuration-File Parameters

Table 7-4 alphabetically lists the DMF configuration file parameters discussed in this chapter.

Table 7-4 Configuration Parameters

Parameter	Section Discussed In
ADMDIR_IN_ROOTFS	"base Object Parameters" on page 278
ADMIN_EMAIL	"base Object Parameters" on page 278
AGE_WEIGHT	"File Weighting Parameters for a Managed Filesystem" on page 371
AGGRESSIVE_HVFY	"drivegroup Object Parameters" on page 397
ALERT_RETENTION	"taskgroup Object Parameters" on page 312
ALGORITHM	"resourcescheduler Object Parameters" on page 432
ALLOCATION_GROUP	"volumegroup Object Parameters" on page 411
ALLOCATION_MAXIMUM	"volumegroup Object Parameters" on page 411
ALLOCATION_MINIMUM	"volumegroup Object Parameters" on page 411
BANDWIDTH_MULTIPLIER	"drivegroup Object Parameters" on page 397
BLOCK_SIZE	"drivegroup Object Parameters" on page 397
BUFFERED_IO_SIZE	"filesystem Object Parameters" on page 351 "msp Object Parameters for a DCM MSP" on page 460
CACHE_AGE_WEIGHT	"File Weighting Parameters for a DCM MSP STORE_DIRECTORY" on page 378
CACHE_DIR	"libraryserver Object Parameters" on page 393
CACHE_MEMBERS	"fastmountcache Object" on page 390
CACHE_SPACE	"libraryserver Object Parameters" on page 393
CACHE_SPACE_WEIGHT	"File Weighting Parameters for a DCM MSP STORE_DIRECTORY" on page 378
CHECKSUM_TYPE	"volumegroup Object Parameters" on page 411

Parameter	Section Discussed In
CHILD_MAXIMUM	"msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455
CLOUD_BACKEND	"msp Object Parameters for a Cloud MSP" on page 449
CLOUD_BASE_PATH	"msp Object Parameters for a Cloud MSP" on page 449
CLOUD_BUCKET	"msp Object Parameters for a Cloud MSP" on page 449
CLOUD_HOST	"msp Object Parameters for a Cloud MSP" on page 449
CLOUD_PASSWORD	"msp Object Parameters for a Cloud MSP" on page 449
CLOUD_REDUNDANCY	"msp Object Parameters for a Cloud MSP" on page 449
CLOUD_USER	"msp Object Parameters for a Cloud MSP" on page 449
COMMAND	"libraryserver Object Parameters" on page 393 "msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455
COMPRESSION_TYPE	"drivegroup Object Parameters" on page 397
COPAN_VSNS	"libraryserver Object Parameters" on page 393
DATA_LIMIT	"taskgroup Object Parameters" on page 312
DATABASE_COPIES	"taskgroup Object Parameters" on page 312
DB_SERVER_BIN	"base Object Parameters" on page 278
DB_SERVER_HOST	"base Object Parameters" on page 278
DB_SERVER_PORT	"base Object Parameters" on page 278
DIRECT_IO_MAXIMUM_SIZE	"base Object Parameters" on page 278
DIRECT_IO_SIZE	"filesystem Object Parameters" on page 351 "msp Object Parameters for a DCM MSP" on page 460
DISCONNECT_TIMEOUT	"libraryserver Object Parameters" on page 393
DMMIGRATE_MINIMUM_AGE	"taskgroup Object Parameters" on page 312
DMMIGRATE_MIN_OFFLINE_AGE	"taskgroup Object Parameters" on page 312
DMMIGRATE_MIN_OFFLINE_SIZE	"taskgroup Object Parameters" on page 312

Parameter	Section Discussed In
DMMIGRATE_QUEUE_MAX	"taskgroup Object Parameters" on page 312
DMMIGRATE_TRICKLE	"taskgroup Object Parameters" on page 312
DMMIGRATE_UNACK	"taskgroup Object Parameters" on page 312
DMMIGRATE_VERBOSE	"taskgroup Object Parameters" on page 312
DMMIGRATE_WAIT	"taskgroup Object Parameters" on page 312
DRIVE_GROUPS	"libraryserver Object Parameters" on page 393
DRIVE_MAXIMUM	"drivegroup Object Parameters" on page 397 "volumegroup Object Parameters" on page 411
DRIVE_SCHEDULER	"drivegroup Object Parameters" on page 397 "volumegroup Object Parameters" on page 411
DRIVES_TO_DOWN	"drivegroup Object Parameters" on page 397
DRIVETAB	"taskgroup Object Parameters" on page 312
DSK_BUFSIZE	"msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467
DUALRESIDENCE_TARGET	"Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376 "msp Object Parameters for a DCM MSP" on page 460
DUMP_COMPRESS	"dumpgroup Object Parameters" on page 343 "filesystem Object Parameters" on page 351 "taskgroup Object Parameters" on page 312
DUMP_CONCURRENCY	"taskgroup Object Parameters" on page 312
DUMP_DESTINATION	"taskgroup Object Parameters" on page 312
DUMP_DEVICE	"taskgroup Object Parameters" on page 312
DUMP_DR_EMAIL	"taskgroup Object Parameters" on page 312
DUMP_DR_EMAIL_MAX_SIZE	"taskgroup Object Parameters" on page 312
DUMP_DR_MINIMAL	"taskgroup Object Parameters" on page 312
DUMP_DR_MIRRORS	"taskgroup Object Parameters" on page 312
DUMP_FILE_SYSTEMS	"dumpgroup Object Parameters" on page 343 "taskgroup Object Parameters" on page 312

Parameter	Section Discussed In
DUMP_FLUSH_DCM_FIRST	"msp Object Parameters for a DCM MSP" on page 460 "taskgroup Object Parameters" on page 312
DUMP_GROUPS	"taskgroup Object Parameters" on page 312
DUMP_INVENTORY_COPY	"taskgroup Object Parameters" on page 312
DUMP_MAX_FILESPACE	"dumpgroup Object Parameters" on page 343 "filesystem Object Parameters" on page 351 "taskgroup Object Parameters" on page 312
DUMP_MIGRATE_FIRST	"dumpgroup Object Parameters" on page 343 "filesystem Object Parameters" on page 351 "taskgroup Object Parameters" on page 312
DUMP_MIRRORS	"taskgroup Object Parameters" on page 312
DUMP_MISC_LOCATIONS	"taskgroup Object Parameters" on page 312
DUMP_POST_BACKUP_SCRIPT	"taskgroup Object Parameters" on page 312
DUMP_POST_DR_SCRIPT	"taskgroup Object Parameters" on page 312
DUMP_PRE_BACKUP_SCRIPT	"taskgroup Object Parameters" on page 312
DUMP_RETENTION	"taskgroup Object Parameters" on page 312
DUMP_STREAMS	"dumpgroup Object Parameters" on page 343 "filesystem Object Parameters" on page 351 "taskgroup Object Parameters" on page 312
DUMP_TAPES	"taskgroup Object Parameters" on page 312
DUMP_VSNS_USED	"taskgroup Object Parameters" on page 312
DUMP_WORKSPACE	"taskgroup Object Parameters" on page 312
DUMP_XFSDUMP_PARAMS	"dumpgroup Object Parameters" on page 343 "filesystem Object Parameters" on page 351 "taskgroup Object Parameters" on page 312
EXPORT_METRICS	"base Object Parameters" on page 278
EXPORT_QUEUE	"dmdaemon Object Parameters" on page 290
FADV_SIZE_MAID	"drivegroup Object Parameters" on page 397
FADV_SIZE_MSP	"msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467

Parameter	Section Discussed In
FILE_RETENTION_DAYS	"taskgroup Object Parameters" on page 312
FMC_MOVEFS	"taskgroup Object Parameters" on page 312
FMC_NAME	"taskgroup Object Parameters" on page 312
FORWARD_RECALLS	"volumegroup Object Parameters" on page 411
FREE_DUALRESIDENT_FIRST	"Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376
FREE_DUALSTATE_FIRST	"Automated Space-Management Parameters for a Managed Filesystem" on page 369
FREE_SPACE_DECREMENT	"Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376 "Automated Space-Management Parameters for a Managed Filesystem" on page 369
FREE_SPACE_MINIMUM	"Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376 "Automated Space-Management Parameters for a Managed Filesystem" on page 369
FREE_SPACE_TARGET	"Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376 "Automated Space-Management Parameters for a Managed Filesystem" on page 369
FREE_VOLUME_MINIMUM	"taskgroup Object Parameters" on page 312
FREE_VOLUME_TARGET	"taskgroup Object Parameters" on page 312
FTP_ACCOUNT	"msp Object Parameters for an FTP MSP" on page 455
FTP_COMMAND	"msp Object Parameters for an FTP MSP" on page 455
FTP_DIRECTORY	"msp Object Parameters for an FTP MSP" on page 455
FTP_HOST	"msp Object Parameters for an FTP MSP" on page 455
FTP_PASSWORD	"msp Object Parameters for an FTP MSP" on page 455
FTP_PORT	"msp Object Parameters for an FTP MSP" on page 455
FTP_USER	"msp Object Parameters for an FTP MSP" on page 455
FULL_THRESHOLD_BYTES	"msp Object Parameters for a Disk MSP" on page 467

Parameter	Section Discussed In
FULL_ZONE	"volumegroup Object Parameters" on page 411
GET_WAIT_TIME	"volumegroup Object Parameters" on page 411
GROUP_MEMBERS	"migrategroup Object Parameters" on page 427
GUARANTEED_DELETES	"msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455
GUARANTEED_GETS	"msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455
HBA_BANDWIDTH	"base Object Parameters" on page 278 "node Object Parameters" on page 294
HFREE_TIME	"volumegroup Object Parameters" on page 411
HOME_DIR	"base Object Parameters" on page 278
HTML_REFRESH	"resourcewatcher Object Parameters" on page 434
IMMUTABLE_BFIDS	"filesystem Object Parameters" on page 351
IMPORT_DELETE	"msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455
IMPORT_ONLY	"msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455 "volumegroup Object Parameters" on page 411
INTERFACE	"node Object Parameters" on page 294
JOURNAL_DIR	"base Object Parameters" on page 278
JOURNAL_RETENTION	"taskgroup Object Parameters" on page 312
JOURNAL_SIZE	"base Object Parameters" on page 278
KILL_WC_IF_DEMAND	"volumegroup Object Parameters" on page 411
LABEL_TYPE	"drivegroup Object Parameters" on page 397
LICENSE_FILE	"base Object Parameters" on page 278
LOG_RETENTION	"taskgroup Object Parameters" on page 312

Parameter	Section Discussed In
LOGICAL_BLOCK_PROTECTION	"volumegroup Object Parameters" on page 411
LS_NAMES	"dmdaemon Object Parameters" on page 290
MAX_ALERTDB_SIZE	"taskgroup Object Parameters" on page 312
MAX_CACHE_FILE	"libraryserver Object Parameters" on page 393
MAX_CHUNK_SIZE	"volumegroup Object Parameters" on page 411
MAX_IDLE_PUT_CHILDREN	"volumegroup Object Parameters" on page 411
MAX_MANAGED_REGIONS	"filesystem Object Parameters" on page 351
MAX_MS_RESTARTS	"drivegroup Object Parameters" on page 397
MAX_PERFDB_SIZE	"taskgroup Object Parameters" on page 312
MAX_PUT_CHILDREN	"drivegroup Object Parameters" on page 397 "volumegroup Object Parameters" on page 411
MERGE_CUTOFF	"volumegroup Object Parameters" on page 411
MERGE_INTERFACE	"node Object Parameters" on page 294"node Object" on page 294
MERGE_THRESHOLD	"volumegroup Object Parameters" on page 411
MESSAGE_LEVEL	"dmdaemon Object Parameters" on page 290 "filesystem Object Parameters" on page 351 "libraryserver Object Parameters" on page 393 "msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455 "services Object Parameters" on page 300 Chapter 10, "Message Log Files" on page 507
METRICS_RETENTION	"base Object Parameters" on page 278
MIGRATION_LEVEL	"dmdaemon Object Parameters" on page 290 "filesystem Object Parameters" on page 351 "msp Object Parameters for a DCM MSP" on page 460
MIGRATION_TARGET	"Automated Space-Management Parameters for a Managed Filesystem" on page 369
MIN_ARCHIVE_SIZE	"filesystem Object Parameters" on page 351

Parameter	Section Discussed In
MIN_DIRECT_SIZE	"msp Object Parameters for a DCM MSP" on page 460 "filesystem Object Parameters" on page 351
MIN_VOLUMES	"volumegroup Object Parameters" on page 411
MOUNT_BLOCKED_TIMEOUT	"drivegroup Object Parameters" on page 397
MOUNT_SERVICE	"device Object Parameters" on page 348 "drivegroup Object Parameters" on page 397
MOUNT_SERVICE_GROUP	"device Object Parameters" on page 348 "drivegroup Object Parameters" on page 397
MOUNT_TIMEOUT	"drivegroup Object Parameters" on page 397
MOVE_FS	"dmdaemon Object Parameters" on page 290
MSG_DELAY	"drivegroup Object Parameters" on page 397
MSP_NAMES	"dmdaemon Object Parameters" on page 290
MULTIPLIER	"migrategroup Object Parameters" on page 427
MULTITAPE_NODES	"drivegroup Object Parameters" on page 397
MVS_UNIT	"msp Object Parameters for an FTP MSP" on page 455
NAME_FORMAT	"msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455
NODE_ANNOUNCE_RATE	"services Object Parameters" on page 300
NODE_BANDWIDTH	"base Object Parameters" on page 278 "node Object Parameters" on page 294
NODE_TIMEOUT	"services Object Parameters" on page 300
OV_ACCESS_MODES	"device Object Parameters" on page 348 "drivegroup Object Parameters" on page 397
OV_INTERCHANGE_MODES	"device Object Parameters" on page 348 "drivegroup Object Parameters" on page 397
OV_KEY_FILE	"base Object Parameters" on page 278
OV_SERVER	"base Object Parameters" on page 278
PARTIAL_STATE_FILES	"dmdaemon Object Parameters" on page 290

Parameter	Section Discussed In
PENALTY	"resourcescheduler Object Parameters" on page 432
PERF_RETENTION	"taskgroup Object Parameters" on page 312
PERFTRACE_METRICS	"base Object Parameters" on page 278
POLICIES	"filesystem Object Parameters" on page 351 "msp Object Parameters for a DCM MSP" on page 460
POSITIONING	"drivegroup Object Parameters" on page 397
POSITION_RETRY	"drivegroup Object Parameters" on page 397
POSIX_FADVISE_SIZE	"filesystem Object Parameters" on page 351
PRIORITY_PERIOD	"msp Object Parameters for a DCM MSP" on page 460
PUT_IDLE_DELAY	"volumegroup Object Parameters" on page 411
PUTS_TIME	"volumegroup Object Parameters" on page 411
READ_ERR_MAXIMUM	"drivegroup Object Parameters" on page 397
READ_ERR_MINIMUM	"drivegroup Object Parameters" on page 397
READ_ERR_TIMEOUT	"drivegroup Object Parameters" on page 397
READ_IDLE_DELAY	"drivegroup Object Parameters" on page 397 "volumegroup Object Parameters" on page 411
READ_TIME	"volumegroup Object Parameters" on page 411
RECALL_NOTIFICATION_RATE	"dmdaemon Object Parameters" on page 290
REDIRECT_TIME_DG_DOWN	"drivegroup Object Parameters" on page 397
REINSTATE_DRIVE_DELAY	"drivegroup Object Parameters" on page 397
REINSTATE_VOLUME_DELAY	"drivegroup Object Parameters" on page 397
RELEASETHREADS	"filesystem Object Parameters" on page 351
REMALERT_PARAMS	"taskgroup Object Parameters" on page 312
REMPERF_PARAMS	"taskgroup Object Parameters" on page 312
RESERVED_VOLUMES	"volumegroup Object Parameters" on page 411
REWIND_DELAY	"drivegroup Object Parameters" on page 397 "volumegroup Object Parameters" on page 411
ROTATION_STRATEGY	"migrategroup Object Parameters" on page 427

Parameter	Section Discussed In
RUN_TASK	"Automated Maintenance Tasks" on page 175 "drivegroup Object Parameters" on page 397 "libraryserver Object Parameters" on page 393 "taskgroup Object Parameters" on page 312 "volumegroup Object Parameters" on page 411
SCAN_FILESYSTEMS	"taskgroup Object Parameters" on page 312
SCAN_FOR_DMSTAT	"taskgroup Object Parameters" on page 312
SCAN_OUTPUT	"taskgroup Object Parameters" on page 312
SCAN_PARALLEL	"taskgroup Object Parameters" on page 312
SCAN_PARAMS	"taskgroup Object Parameters" on page 312
SELECT_LOWER_VG	"VG Selection Parameters for a DCM MSP STORE_DIRECTORY" on page 380
SELECT_MSP	"MSP / VG Selection Parameters for a Managed Filesystem" on page 374
SELECT_VG	"MSP / VG Selection Parameters for a Managed Filesystem" on page 374
SERVER_NAME	"base Object Parameters" on page 278
SERVICES	"node Object Parameters" on page 294
SERVICES_PORT	"services Object Parameters" on page 300
SITE_SCRIPT	"Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376 "Automated Space-Management Parameters for a Managed Filesystem" on page 369 "msp Object Parameters for a DCM MSP" on page 460
SPACE_WEIGHT	"File Weighting Parameters for a Managed Filesystem" on page 371
SPOOL_DIR	"base Object Parameters" on page 278
STORE_DIRECTORY	"msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467

Parameter	Section Discussed In
TASK_GROUPS	"dmdaemon Object Parameters" on page 290 "drivegroup Object Parameters" on page 397 "filesystem Object Parameters" on page 351 "libraryserver Object Parameters" on page 393 "msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455 "services Object Parameters" on page 300 "taskgroup Object Parameters" on page 312 "volumegroup Object Parameters" on page 411
THRESHOLD	"taskgroup Object Parameters" on page 312
TIMEOUT_FLUSH	"volumegroup Object Parameters" on page 411
TMF_TMMNT_OPTIONS	"device Object Parameters" on page 348 "drivegroup Object Parameters" on page 397
TMP_DIR	"base Object Parameters" on page 278
TSREPORT_OPTIONS	"taskgroup Object Parameters" on page 312

Parameter	Section Discussed In
TYPE	"allocationgroup Object Parameters" on page 435 "base Object Parameters" on page 278 "device Object Parameters" on page 348 "dmdaemon Object Parameters" on page 290 "drivegroup Object Parameters" on page 397 "dumpgroup Object Parameters" on page 343 "fastmountcache Object" on page 390 "filesystem Object Parameters" on page 351 "libraryserver Object Parameters" on page 393 "migrategroup Object Parameters" on page 427 "msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455 "node Object Parameters" on page 294 "policy Object" on page 364 "resourcescheduler Object Parameters" on page 432 "resourcewatcher Object Parameters" on page 434 "services Object Parameters" on page 300 "taskgroup Object Parameters" on page 312 "volumegroup Object Parameters" on page 411
UPDATE_CTIME	"filesystem Object Parameters" on page 351
USE_AS_DUMP_DEST	"filesystem Object Parameters" on page 351
USE_UNIFIED_BUFFER	"filesystem Object Parameters" on page 351
VALID_ROOT_HOSTS	"base Object Parameters" on page 278
VERIFY_POSITION	"drivegroup Object Parameters" on page 397
VOL_MSG_TIME	"allocationgroup Object Parameters" on page 435 "volumegroup Object Parameters" on page 411
VOLUME_GROUPS	"drivegroup Object Parameters" on page 397
VOLUME_LIMIT	"taskgroup Object Parameters" on page 312
WATCHER	"libraryserver Object Parameters" on page 393
WEIGHT	"resourcescheduler Object Parameters" on page 432

Parameter	Section Discussed In
WRITE_CHECKSUM	"drivegroup Object Parameters" on page 397 "msp Object Parameters for a Cloud MSP" on page 449 "msp Object Parameters for a DCM MSP" on page 460 "msp Object Parameters for a Disk MSP" on page 467 "msp Object Parameters for an FTP MSP" on page 455
ZONE_SIZE	"volumegroup Object Parameters" on page 411

Parallel DMF Configuration

This chapter discusses the following:

- "Parallel DMF Configuration Procedure" on page 485
- "Determining the State of Parallel Data-Mover Nodes" on page 488
- "Disabling Parallel Data-Mover Nodes" on page 489
- "Reenabling Parallel Data-Mover Nodes" on page 489

Parallel DMF Configuration Procedure

If you are running DMF with the Parallel Data-Mover Option (Parallel DMF), do the following:

Procedure 8-1 Configuring Parallel DMF

1. Configure the DMF configuration file (`/etc/dmf/dmf.conf`) on the DMF server according to the instructions in " Object Definitions" on page 272. Ensure that a `node` object is defined in `dmf.conf` for the parallel data-mover node that is being added.
2. Copy `/etc/dmf/dmf.conf` on the DMF server to `/etc/dmf/dmf.conf` on the DMF parallel data-mover node.

   Note: Do not edit the `dmf.conf` file on the parallel data-mover node.

3. Install the software for the parallel data mover on the parallel data-mover node. See the *SGI InfiniteStorage Software Platform* release note for more information.
4. Configure CXFS according to the instructions in the *CXFS 7 Administrator Guide for SGI InfiniteStorage.*
5. Include the DMF parallel data-mover node as a CXFS client, such as by creating an `autoconf` rule. For more information, see the section about the `autoconf` command in the `cxfs_admin` chapter of the CXFS administrator guide or the `cxfs_admin`(8) man page.

For example, for two parallel data-mover nodes named pdm1 and pdm2 in a CXFS cluster named mycluster:

```
# cxfs_admin -c "create autoconf rule_name=pdm1rule policy=allowed \
hostname=pdm1 enable_node=true" -i mycluster
# cxfs_admin -c "create autoconf rule_name=pdm2rule policy=allowed \
hostname=pdm2 enable_node=true" -i mycluster
```

After you have finished creating or modifying all of the desired autoconf rules, you must unlock all cxfs_admin sessions in order for nodes to be automatically configured. (The automatic configuration process must have access to the cxfs_admin lock.)

If a node you refer to in an autoconf rule has previously been part of the CXFS cluster, or if the node fails to join the CXFS membership, you must reboot the node.

6. Configure the CXFS filesystems defined by the following DMF configuration parameters so that they are mounted only on the primary DMF server, the passive DMF server (if applicable), and each parallel data-mover node:

   HOME_DIR
   CACHE_DIR
   MOVE_FS
   TMP_DIR
   SPOOL_DIR
   STORE_DIRECTORY for a DCM MSP

   For more information about these parameters, see "base Object" on page 277.

   For example, if the filesystem to be mounted on the directory specified by CACHE_DIR is on the /dev/cxvm/fscache device, you could specify the following cxfs_admin commands to restrict it to the CXFS potential metadata server nodes on which the DMF server can run (say server1 and server2) and the parallel data-mover nodes (say pdm1 and pdm2):

   ```
   # cxfs_admin -c "create filesystem name=fscache mount_new_nodes=false \
   nodes=server1,server2,pdm1,pdm2" -i mycluster
   ```

   For more information, see the section about the mount command in the cxfs_admin chapter of the CXFS administrator guide.

7. If you use a directory other than those listed in step 6 for the OV_KEY_FILE configuration parameter, ensure that the OpenVault security key file is visible to the DMF server and all parallel data-mover nodes. See "base Object" on page 277.
8. Configure the managed filesystems as CXFS filesystems that are mounted on the DMF server and all of the parallel data-mover nodes. They may also be mounted on CXFS client-only nodes.
9. On the DMF server, use ov_admin to allow the parallel data-mover node to be a DCP-enabled OpenVault client machine. Do the following:

   a. From the main menu in ov_admin, enter 22 to select Manage OpenVault Client Machines.

   b. Enter 1 to select Activate an OpenVault Client Machine and follow the prompts. Be sure to answer yes when asked if the machine will run DCPs.

   For more information about ov_admin, see the *OpenVault Administrator Guide for SGI InfiniteStorage*.
10. On the parallel data-mover node, use ov_admin to configure DCPs for those drives that it should operate.
11. If not already done, activate a privileged instance and an unprivileged instance of the dmf application for each parallel data-mover node. See "Configure OpenVault for DMF Use" on page 494.
12. Verify the DMF configuration; if there are errors, fix them and repeat the verification until there are no errors. You can do this by using DMF Manager or the dmcheck(8) script on the DMF server. For more information, see Chapter 6, "DMF Manager" on page 191.
13. Start the DMF mover service on the parallel data-mover node:

    ```
    mover# service dmf_mover start
    ```

After initial configuration, changes to dmf.conf will normally be propagated to parallel data-mover nodes automatically while the DMF services are running. Certain changes, such as changing the SERVER_NAME or SERVICES_PORT of the DMF server, will require that you manually copy dmf.conf to the parallel data-mover nodes and then restart the DMF services on those nodes.

Determining the State of Parallel Data-Mover Nodes

To determine the status of a parallel data-mover node, enter the following command as root:

```
# dmnode_admin -l
```

For example, showing the state for parallel data-mover nodes jar and zin:

```
# dmnode_admin -l
Node Name  State     Enabled  Active Since          Dropouts
jar        Inactive  Yes      -                     0
zin        Active    Yes      2008-Nov-26,12:45:48  0
```

The node state can be one of the following:

Active	The node is connected to dmnode_service on the DMF server and is eligible to run data-mover processes.
Inactive	The node is not connected to dmnode_service.
Disabled	The node is connected to dmnode_service but has been disabled using dmnode_admin. See "Disabling Parallel Data-Mover Nodes" on page 489.
FileSysWait	The node is connected to dmnode_service but has not been made active by dmnode_admin because a filesystem is mounted on the server but not on the node. The filesystem could be a managed filesystem or one of the filesystems identified by the following parameters in the DMF configuration file: CACHE_DIR HOME_DIR (if it contains the file specified by OV_KEY_FILE) MOVE_FS SPOOL_DIR STORE_DIRECTORY for a DCM MSP TMP_DIR
License Wait	The node is connected to dmnode_service but has not been made active by dmnode_admin due to the

lack of a sufficient number of DMF parallel data mover licenses on the server.

The `Dropouts` field specifies the number of times that the node has transitioned from `Active` to `Inactive`. A non-zero count may indicate a problem with the mover node or network. This count is reset when `dmnode_service` is restarted.

Note: If `dmnode_service` is not running, the `dmnode_admin` command will not function. To restart `dmnode_service`, enter the following:

```
# service dmf start
```

Disabling Parallel Data-Mover Nodes

To disable parallel data-mover nodes in order to perform maintenance on the system or to diagnose a problem, enter the following:

```
# dmnode_admin -d nodename ...
```

The node will remain disabled across DMF restarts.

The disabled node is no longer eligible to start new data-mover processes.

Existing data-mover processes on the disabled node will be told to exit after the library server notices this change, which may take up to 2 minutes. The existing data-mover processes may exit in the middle of recalling or migrating a file; this work will be reassigned to other data-mover processes. Stopping data-mover processes with the following command has the same result on existing processes:

```
# service dmf_mover stop
```

Reenabling Parallel Data-Mover Nodes

To reenable parallel data-mover nodes, making them eligible to run data-mover processes, enter the following as `root`:

```
# dmnode_admin -e nodename ...
```

The node will remain enabled across DMF restarts.

To determine the current state of a node, see "Determining the State of Parallel Data-Mover Nodes" on page 488.

Note: DMF and DMF Manager must be running for the `dmnode_admin` command to function.

Mounting Service Configuration Tasks

This chapter discusses the following:

- "OpenVault Configuration Tasks" on page 491
- "TMF Configuration Tasks" on page 505

OpenVault Configuration Tasks

This section discusses the following:

- "Configure the OpenVault Server" on page 492
- "Configure OpenVault for DMF Use" on page 494
- "Configure OpenVault for Each Parallel Data-Mover Node" on page 498
- "Configure OpenVault on the DMF Server If on a Different Host" on page 502
- "Configure OpenVault for a Drive Group" on page 502

Note: For additional information about COPAN MAID or COPAN VTL and OpenVault, see:

- *COPAN MAID for DMF Quick Start Guide*
- *SGI 400 VTL for DMF Quick Start Guide*

Configure the OpenVault Server

Following is an example of the steps you will take to initially configure the OpenVault server, using an example host named `dmfserver` as the OpenVault server (typically, the same host will be the DMF server and the OpenVault server). The characters **###** in the right margin highlight comments related to the steps, which follow the example:

```
dmfserver# ov_admin                                                    ### Step 1

    OpenVault Configuration

The general strategy for setting up OpenVault is to

     1) configure the OpenVault server
     2) configure LCP/DCPs on the server machine
     3) configure server for local Applications
     4) if needed, configure server for remote LCPs, DCPs, and Applications
     5) if needed, install and configure LCP/DCPs on remote machines
     6) from the server, for each library setup/import media

Where possible, defaults for each prompt are indicated by [value].
Help text may be obtained by entering '?' at most prompts.

Some menus will present only the available options depending
upon the software, hardware, or options that are installed.
If you do not see the choice you are looking for, double check
your installation to make sure the items are installed.

Press enter to continue...

There may be multiple OpenVault servers and networks in your area.
Enter the name where the OpenVault server is listening (or will be
listening after it has been configured). This may be the server's
system hostname, or the hostname of another interface on the server
if an alternative network is being used.

Name where the OpenVault server is (or will be) listening? [dmfserver]     ### Step 2

The OpenVault server is not yet configured; would you like to do so now? [Yes]    ### Step 3

What port number should the OpenVault server use? [695]                    ### Step 4
```

```
What security key would you like the admin commands to use? [none]                    ### Step 5
Waiting for OpenVault to initialize ...
The OpenVault server was successfully started.
```

Comments:

1. Log in to the system where the OpenVault server will run and invoke the OpenVault administration tool `ov_admin`(8). SGI recommends that the OpenVault server run on the same node as the DMF server.
2. Enter the name associated with the IP address where the OpenVault server will listen. If OpenVault will be running on the same server as DMF, the OpenVault server should listen on the same interface used for DMF communications. Enter:
   - The server's virtual hostname if using high availability (HA)
   - The hostname used for the server's `INTERFACE` parameter (see "`node` Object" on page 294) if using the Parallel Data-Mover option with an alternative network.
   - The system hostname if using basic DMF or the Parallel Data-Mover Option with the default interface

   Note: You must set the `OV_SERVER` parameter in the `base` object (see "`base` Object" on page 277).
3. Enter `Yes` to configure the OpenVault server.
4. Select a port number for the OpenVault server. Normally, you can use the default. You must use the same port number when configuring OpenVault on any parallel data-mover nodes.
5. Optionally provide a security key to prevent unauthorized clients from using the OpenVault administration commands if your environment is not secure (this will result in larger log files.)

Configure OpenVault for DMF Use

You must give DMF permission to connect to OpenVault from various hosts and make use of drives and volumes by activating instances of the `dmf` application. The following example uses a host named `dmfserver` as the OpenVault server:

1. Add an unprivileged instance and then an privileged instance of the `dmf` application:

```
dmfserver# ov_admin                                                    ### Step a

Name where the OpenVault server is listening? [dmfserver]              ### Step b

    OpenVault Configuration Menu for server "dmfserver"

      Configuration on Machines Running LCPs and DCPs
         1 - Manage LCPs for locally attached Libraries
         2 - Manage DCPs for locally attached Drives

      Configuration on Admin-Enabled Machines
        11 - Manage Cartridge Groups
        12 - Manage Drive Groups
        13 - Import Media

      Configuration on the OpenVault Server Machine
        21 - Manage Applications
        22 - Manage OpenVault Client Machines

         q - Exit.

    Which operation would you like to do: 21                           ### Step c


    Manage Applications Menu

         1 - Create a new Application

         2 - Delete an Application
         3 - Show all existing Applications

         4 - Activate another Application Instance for an existing Application
         5 - Deactivate an Application Instance
```

```
        6 - Show all activated Application Instances

        r - Return to Main Menu.
        q - Exit.


    Which operation would you like to do: 4                                    ### Step d


    Select the Application for which you want to activate a new Instance

        1 - dmf
        2 - ov_umsh

        r - Return to Previous Menu.
        q - Exit.

    Which operation would you like to do: 1                                    ### Step e

Enter the name of the Host where an instance of Application "dmf" will run [dmfserver] *  ### Step f

Enter the Application's instance name or "*" [] *                              ### Step g

Should this Instance of the Application "dmf" be "privileged"? [No]            ### Step h

What security key will the Application use [none]                              ### Step i

Unprivileged Instance "*" of Application "dmf"
was successfully activated on "dmfserver".

Press enter to continue...                                                     ### Step j

    Manage Applications Menu

        1 - Create a new Application
        2 - Delete an Application
        3 - Show all existing Applications

        4 - Activate another Application Instance for an existing Application
        5 - Deactivate an Application Instance
        6 - Show all activated Application Instances
```

```
        r - Return to Main Menu.
        q - Exit.


    Which operation would you like to do: 4                                    ### Step k


    Select the Application for which you want to activate a new Instance

        1 - dmf
        2 - ov_umsh

        r - Return to Previous Menu.
        q - Exit.

    Which operation would you like to do: 1                                    ### Step l

Enter the name of the Host where an instance of Application "dmf" will run [dmfserver] *  ### Step m

Enter the Application's instance name or "*" [] *                              ### Step n

Should this Instance of the Application "dmf" be "privileged"? [No] yes        ### Step o

What security key will the Application use [none]                              ### Step p

Privileged Instance "*" of Application "dmf"
was successfully activated on "dmfserver".

Press enter to continue...

    Manage Applications Menu

        1 - Create a new Application
        2 - Delete an Application
        3 - Show all existing Applications

        4 - Activate another Application Instance for an existing Application
        5 - Deactivate an Application Instance
        6 - Show all activated Application Instances
```

```
        r - Return to Main Menu.
        q - Exit.


    Which operation would you like to do: q                                    ### Step q
dmfserver#
```

Comments:

a. Log in to the OpenVault server and invoke the OpenVault administration tool `ov_admin`(8).

b. Enter the name associated with the IP address on which the OpenVault server is listening.

c. Enter `21` to manage applications.

d. Enter `4` to activate another application instance.

e. Enter `1` to select the application `dmf`.

f. Enter the wildcard `*` character to allow the `dmf` application to be used from any host. (Alternatively, you can repeat these steps to create a privileged and an unprivileged application instance for each system that DMF runs on; that is, each DMF server and each parallel data-mover node).

g. Enter the wildcard `*` for the application instance name.

h. Use the default (`No`) to create the unprivileged instance.

i. Optionally provide a security key.

j. Press `Enter` to continue.

k. Enter `4` to activate another application instance.

l. Enter `1` to select the application `dmf`.

m. Enter the wildcard `*` character to allow the `dmf` application to be used from any host. (See step f.)

n. Enter the wildcard `*` for the application instance name.

o. Enter `yes` to make the application privileged.

p. Optionally provide a security key.

q. Enter `q` to exit.

2. Configure the `base` object for use with OpenVault. For example:

```
define  base
        TYPE            base
        HOME_DIR        /dmf/home
.
.
.
        OV_KEY_FILE     /dmf/home/ov_keys
```

   For more information, see "`base` Object" on page 277.

3. Use the `dmov_keyfile`(8) command to create the file defined by the `OV_KEY_FILE` parameter. This command will prompt you for the unprivileged and privileged keys that you defined.

Configure OpenVault for Each Parallel Data-Mover Node

Following is an example of the steps you will take to configure the Parallel Data-Mover Option. You will repeat these steps for each parallel data-mover node:

1. On the OpenVault server (for example, named `dmfserver`), activate the parallel data-mover node (such as `mover1`) as a client:

```
dmfserver# ov_admin                                                    ### Step 1a

Name where the OpenVault server is (or will be) listening? [dmfserver]     ### Step 1b

     OpenVault Configuration Menu for server "dmfserver"

       Configuration on Machines Running LCPs and DCPs
          1 - Manage LCPs for locally attached Libraries
          2 - Manage DCPs for locally attached Drives

       Configuration on Admin-Enabled Machines
         11 - Manage Cartridge Groups
         12 - Manage Drive Groups
         13 - Import Media

       Configuration on the OpenVault Server Machine
```

```
         21 - Manage Applications
         22 - Manage OpenVault Client Machines

          q - Exit.

     Which operation would you like to do: 22                              ### Step 1c


     Manage OpenVault Client Machines Menu

          1 - Activate an OpenVault Client Machine
          2 - Deactivate an OpenVault Client Machine
          3 - Show all OpenVault Client Machines

          r - Return to Main Menu.
          q - Exit.


     Which operation would you like to do: 1                               ### Step 1d


Which Client Machine do you want to activate? [] mover1                    ### Step 1e

What security key would you like the Client Machine mover1 to use? [none]  ### Step 1f

Will DCPs and/or LCPs also be configured to run on "mover1"? [Yes]         ### Step 1g

The Client Machine "mover1" was successfully activated.

Press enter to continue...

     Manage OpenVault Client Machines Menu

          1 - Activate an OpenVault Client Machine
          2 - Deactivate an OpenVault Client Machine
          3 - Show all OpenVault Client Machines

          r - Return to Main Menu.
          q - Exit.

     Which operation would you like to do: q                               ### Step 1h
```

Comments:

a. Log in to the OpenVault server and invoke the OpenVault administration tool `ov_admin`(8).

b. Enter the name associated with the IP address on which the OpenVault server is listening.

c. Enter `22` to manage an OpenVault client.

d. Enter `1` to activate a client.

e. Enter the system name of the parallel data-mover node, such as `mover1`.

f. Optionally provide a security key to protect against clients masquerading as allowed clients.

g. Press `Enter` to allow DCPs and LCPs to run on the parallel data-mover node.

h. Enter `q` to exit.

2. On the parallel data-mover node (for example, `mover1`), specify the name on which OpenVault is listening, the port number, and optional security key:

```
mover1# ov_admin                                                    ### Step 2a

     OpenVault Configuration

The general strategy for setting up OpenVault is to

      1) configure the OpenVault server
      2) configure LCP/DCPs on the server machine
      3) configure server for local Applications
      4) if needed, configure server for remote LCPs, DCPs, and Applications
      5) if needed, install and configure LCP/DCPs on remote machines
      6) from the server, for each library setup/import media

Where possible, defaults for each prompt are indicated by [value].
Help text may be obtained by entering '?' at most prompts.

Some menus will present only the available options depending
upon the software, hardware, or options that are installed.
If you do not see the choice you are looking for, double check
your installation to make sure the items are installed.
```

```
Press enter to continue...

There may be multiple OpenVault servers and networks in your area.
Enter the name where the OpenVault server is listening (or will be
listening after it has been configured). This may be the server's
system hostname, or the hostname of another interface on the server
if an alternative network is being used.

Name where the OpenVault server is (or will be) listening? [dmfserver]     ### Step 2b

What port number is the OpenVault server on dmfserver using? [695]         ### Step 2c

What security key would you like the admin commands to use? [none]         ### Step 2d

     OpenVault Configuration Menu for server "dmfserver"

       Configuration on Machines Running LCPs and DCPs
          1 - Manage LCPs for locally attached Libraries
          2 - Manage DCPs for locally attached Drives

       Configuration on Admin-Enabled Machines
         11 - Manage Cartridge Groups
         12 - Manage Drive Groups
         13 - Import Media

          q - Exit.

     Which operation would you like to do: q                                ### Step 2e
```

Comments:

a. Log in to the parallel data-mover node and invoke the OpenVault administration tool `ov_admin`(8).

b. Enter the same name here as you did when initially configuring the OpenVault server. (This will also be the same value you entered in 1b).

c. Enter the same port here as you did when initially configuring the OpenVault server (step 4 of "Configure the OpenVault Server" on page 492).

d. If you specified a security key in 1f, enter the same value here.

e. Enter `q` to exit.

Configure OpenVault on the DMF Server If on a Different Host

Note: If the same host is both the OpenVault server and the DMF server, this procedure is not needed.

If the OpenVault server is on a different host from the DMF server, you must repeat the steps in "Configure OpenVault for Each Parallel Data-Mover Node" on page 498 on the DMF server host in order to configure it for OpenVault.

Configure OpenVault for a Drive Group

Procedure 9-1 describes the steps you must take to configure OpenVault for a drive group.

Procedure 9-1 Configuring OpenVault for a Drive Group

Note: The procedure that follows assumes that before you complete the steps described, the OpenVault server is configured and all drives and libraries are configured and OpenVault is running. For more information about configuring OpenVault, see the `ov_admin`(8) man page and *OpenVault Administrator Guide for SGI InfiniteStorage*.

1. Add DMF as a valid application to appropriate cartridge groups.

   The `ov_admin` script allows you to specify the cartridge groups when the DMF application is created or, after creation of the DMF application, you can choose the menu option that allows you to manage cartridge groups. For more information, see the `ov_admin`(8) man page.

2. Add the DMF application as a valid user to appropriate OpenVault drive groups. The OpenVault drive groups that DMF uses must contain only fungible drives. That is, the drives in the OpenVault drive group must have identical characteristics and accessibility, so that any volume that can be mounted and written on one of the drives can also be mounted and read on any of the other drives within the group. Failure to provide identical mounting and accessibility characteristics to all drives in an OpenVault drive group used by an LS might result in mount failures.

Choose the appropriate item from the ov_admin menu. If for some reason you cannot use the ov_admin script, you can enter the command manually, as follows:

```
ov_drivegroup -a -G drive_group -A dmf
```

3. Configure the following parameters as needed in the LS's drivegroup object for use with OpenVault:

```
MOUNT_SERVICE
MOUNT_SERVICE_GROUP
OV_INTERCHANGE_MODES
```

For example:

```
define  dg_c00
        TYPE                    drivegroup
        VOLUME_GROUPS           vg_c00
        MOUNT_SERVICE           openvault
        MOUNT_SERVICE_GROUP     dg_c00
        OV_INTERCHANGE_MODES    compression
enddef
```

For more information, see:

- "device Object" on page 347
- "drivegroup Object Parameters" on page 397

4. Make the appropriate cartridges accessible to the allocation groups, VGs, or filesystem backup scripts by assigning the cartridges to the DMF application in OpenVault. Do the following:

- To find out which drives are in each drive group:

```
# ov_dumptable -n -d'|' -c DriveGroupName,DriveName,LibraryName DRIVE
ultrium3grp|drive1|lib1
ultrium3grp|drive2|lib1
ultrium4grp|drive3|lib1
ultrium4grp|drive4|lib1
```

- To find out which cartridge types each drive can mount:

```
# ov_dumptable -n -d'|' -c DriveName,CartridgeTypeName DCPCAPABILITY | sort -u
drive1|Ultrium1-100
drive1|Ultrium2-200
```

```
drive1|Ultrium3-400
drive2|Ultrium1-100
drive2|Ultrium2-200
drive2|Ultrium3-400
drive3|Ultrium2-200
drive3|Ultrium3-400
drive3|Ultrium4-800
drive4|Ultrium2-200
drive4|Ultrium3-400
drive4|Ultrium4-800
```

In this example, any Ultrium4–800 cartridges can only be used in the `ultrium4grp` drive group.

- To find out the possible cartridge groups:

  # **`ov_cartgroup -s -A dmf`**

- Do one of the following to make both DMF and OpenVault aware of the cartridges to be mounted:

Caution: All cartridges that DMF mounts via OpenVault must have the correct cartridge type. Failure to correctly specify the cartridge type can result in errors when reading and writing data. Contact your SGI service representative if you have questions about cartridge type specification.

  – If you already have tapes defined in your LS database or in a `DUMP_TAPES` file but OpenVault is not aware of them, and every cartridge in the given LS, VG, or task group is of the same cartridge type, you can tell OpenVault about these tapes by entering one of the following:

    `dmov_makecarts [-g` *cartgroup*`] [-t` *carttype*`]` *taskgroupnames*
    `dmov_makecarts [-g` *cartgroup*`] [-t` *carttype*`]` *lsnames*
    `dmov_makecarts [-g` *cartgroup*`] [-t` *carttype*`] [-v` *vg1,vg2,...*`]` *lsname*

    You can replace any of the references to a VG previously mentioned with an AG. If the `-v` parameter is omitted, all VGs and allocation groups in the specified LS will be processed. Tapes will be added to the file controlling the `run_full_dump.sh` and `run_partial_dump.sh` scripts by specifying the name of the task group that refers to them.

– If you have volumes that neither DMF nor OpenVault is aware of, you can import them by cartridge type into OpenVault and add them to DMF by VG, AG, or task group by entering one of the following:

```
dmov_loadtapes [-g cartgroup] [-l library] [-s tapesize] [-t carttype] vgname
dmov_loadtapes [-g cartgroup] [-l library] [-s tapesize] [-t carttype] agname
dmov_loadtapes [-g cartgroup] [-l library] [-s tapesize] [-t carttype] taskgroupname
```

This command will invoke a `vi`(1) session. In the `vi` session, delete any cartridges that you do **not** want added to the LS database. All cartridges that are left in the `vi` session file must be of the same cartridge type, the type you specified with the `-t` option. Volumes will be added to the file controlling the `run_full_dump.sh` and `run_partial_dump.sh` scripts by specifying the name of the task group which refers to them.

– If neither of the above cases apply, you can manually configure the cartridges. The following commands can be useful in this effort:

• Use `ov_stat` to list cartridges in a library. For example:

```
ov_stat -s -L library
```

• Use `ov_lscarts` to list information on cartridges known to OpenVault. For example:

```
ov_lscarts -f '.*'
```

• Use `ov_import` and `dmvoladm` to add the unmanaged cartridges to OpenVault and DMF, and use `vi` to edit the task group in the file specified by the `DUMP_TAPES` parameter in the `taskgroup` stanza in the `dmf.conf` file.

TMF Configuration Tasks

Use one of the following `dmvoladm`(8) commands to add tapes to the LS database:

```
dmvoladm  -l lsname  -c 'create vsn001-vsn010 vg vgname [ts tapesize]'
dmvoladm  -l lsname  -c 'create vsn001-vsn010 vg vgname [ts tapesize]'
```

An AG is specified by the `vg` option, just like a VG. Specifying the tape size will allow commands such as `dmcapacity`(8) and its display in DMF Manager to accurately estimate the remaining capacity of the volume.

There is no special procedure to inform TMF of a tape's existence. TMF assumes that every tape it deals with is in the library or can be provided by an operator, as needed.

Message Log Files

The `dmfdaemon`, `dmlockmgr`, `dmfsmon`, media-specific process (MSP), and library server (LS) message log files use the same general naming convention and message format. The filenames for message logs are created using the extension *yyyymmdd*, which represents the year, month, and day of file creation.

Each line in a message log file begins with the time the message was issued, an optional message level, the process ID number, and the name of the program that issued the message.

The optional message level is described below. The remainder of the line contains informative or diagnostic information. The following sections provide details about each of these logs:

- "Automated Space Management Log File" on page 513 for information about `dmfsmon` and `autolog.`*yyyymmdd*
- "Daemon Logs and Journals" on page 525 for information about `dmfdaemon` and `dmdlog.`*yyyymmdd*
- "`dmlockmgr` Communication and Log Files" on page 527 for information about `dmlockmgr` and `dmlocklog.`*yyyymmdd*
- "LS Logs" on page 539 and "FTP MSP Activity Log" on page 575 for information about `dmatls`, `dmdskmsp`, `dmftpmsp`, and `msplog.`*yyyymmdd*
- Chapter 15, "DMF Maintenance" on page 585, for information about log maintenance

Messages in the `dmdlog`, `dmlocklog`, `moverlog`, and `msplog` files contain a 2–character field immediately following the time field in each message that is issued. This feature helps to categorize the messages and can be used to extract error messages automatically from these logs. Because the only indication of DMF operational failure may be messages written to the DMF logs, recurring problems can go undetected if you do not check the logs daily.

Possible message types for `autolog`, `dmdlog`, `moverlog`, `msplog`, and `dmlocklog` are defined in Table 10-1. The table also lists the corresponding message levels in the configuration file.

Table 10-1 Message Types and Levels

Field	Message Type	Message Level
`-E`	Error	0
`-O`	Ordinary	0
`-I`	Informative	1
`-V`	Verbose	2
`-1`	Debug level 1	3
`-2`	Debug level 2	4
`-3`	Debug level 3	5
`-4`	Debug level 4	6

Automated Space Management

This chapter discusses the following:

- "The dmfsmon Daemon and dmfsfree Command" on page 509
- "Generating the Candidate List" on page 510
- "Selection of Migration Candidates" on page 511
- "Space Management and the DCM MSP" on page 513
- "Automated Space Management Log File" on page 513

The dmfsmon Daemon and dmfsfree Command

The dmfsmon(8) daemon monitors the free-space levels in filesystems configured with automated space management enabled (auto). When the free space in one of the filesystems falls below the free-space minimum, dmfsmon invokes dmfsfree(8). The dmfsfree command attempts to bring the free space and migrated space of a filesystem into compliance with configured values. You can also invoke dmfsfree directly.

When the free space in one of the filesystems falls below its minimum, dmfsfree performs the following steps:

- Scans the filesystem for files that can be migrated and freed or ranges of files that can be freed. Each of these candidates is assigned a weight. This information is used to create a list, called a *candidate list*, that contains an entry for each file or range and is ordered by weight (largest to smallest).
- Selects enough candidates to bring the free space back up to the desired level. Files or ranges of files are selected in order from largest weight to smallest.
- Selects enough regular files from the candidate list to achieve the *migration target*, the integer percentage of total filesystem space that dmfsmon tries to maintain as a reserve of space that is free or occupied by dual-state files (whose online space can be freed quickly) if free space reaches or falls below the *free-space minimum threshold*. Files are selected from the candidate list in order from largest weight to smallest weight.

The dmfsmon daemon should be running whenever DMF is active. You control automated space management by setting the filesystem and policy configuration parameters in the DMF configuration file. The configuration parameters specify targets for migration and free space as well as one or more policies for weighting. Only filesystems configured as MIGRATION_LEVEL auto in the configuration file are included in the space-management process. "policy Object" on page 364, describes how to configure automated space management.

You can change the migration level of a filesystem by editing the configuration file.

Generating the Candidate List

The first step in the migration process occurs when dmfsmon determines it is time to invoke dmfsfree, which scans the filesystem and generates the candidate list. During candidate list generation, the inode of each online file in the specified filesystem is audited and a weight is computed for it.

A filesystem is associated with a weighting policy in the DMF configuration file. The applicable weighting policy determines a file's total weight, or, if a ranges clause is specified in the configuration file, the range's total weight. Total file or range weight is the sum of the AGE_WEIGHT and SPACE_WEIGHT parameters. Defaults are provided for these parameters, and you can configure either to make a change. You do not need to configure a weighting policy if the defaults are acceptable, but you should be aware that the default selects files based on age and not on size. If you want to configure a policy based on size that ignores file age, you should set AGE_WEIGHT to 0 0

The default weighting policy bases the weight of the file on the time that has passed since the file was last accessed or modified. Usually, the more recent a file's access, the more likely it is to be accessed again.

The candidate list is ordered by total file or range weight (largest to smallest). You can prevent a file from being automatically migrated by making sure that no ranges within the file have a positive weight value. You can configure the weighting parameters to have a negative value to ensure that certain files or ranges are never automatically freed.

Note: If you use negative weights to exclude files or ranges from migration, you must ensure that a filesystem does not fill with files or ranges that are never selected for automatic migration.

You can use the dmscanfs(8) command to print file information to standard output (stdout).

Selection of Migration Candidates

The dmfsfree(8) utility processes each ordered candidate list sequentially, seeking candidates to migrate and possibly free. The extent of the selection process is governed by values defined for the filesystem in the DMF configuration file as described in "policy Object" on page 364.

The most essential parameters are as follows:

- FREE_SPACE_MINIMUM
- FREE_SPACE_TARGET
- MIGRATION_TARGET

For more information about these parameters, see:

- "Automated Space-Management Parameters for a Managed Filesystem" on page 369
- "Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376

When dmfsmon detects that the free space on a filesystem has fallen below the level you have set as FREE_SPACE_MINIMUM, it invokes dmfsfree to select a sufficient number of candidates to meet the FREE_SPACE_TARGET. The dmfsfree utility ensures that these files are migrated and releases their disk blocks. It then selects additional candidates to meet the MIGRATION_TARGET and migrates them.

Figure 11-1 shows the relationship of automated space management migration targets to each other. Migration events occur when file activity causes free filesystem space to drop below FREE_SPACE_MINIMUM. dmfsmon generates a candidate list and begins to migrate files and free the disk blocks until the FREE_SPACE_TARGET is met, and then it migrates regular files (creating dual-state files) until the MIGRATION_TARGET is met.

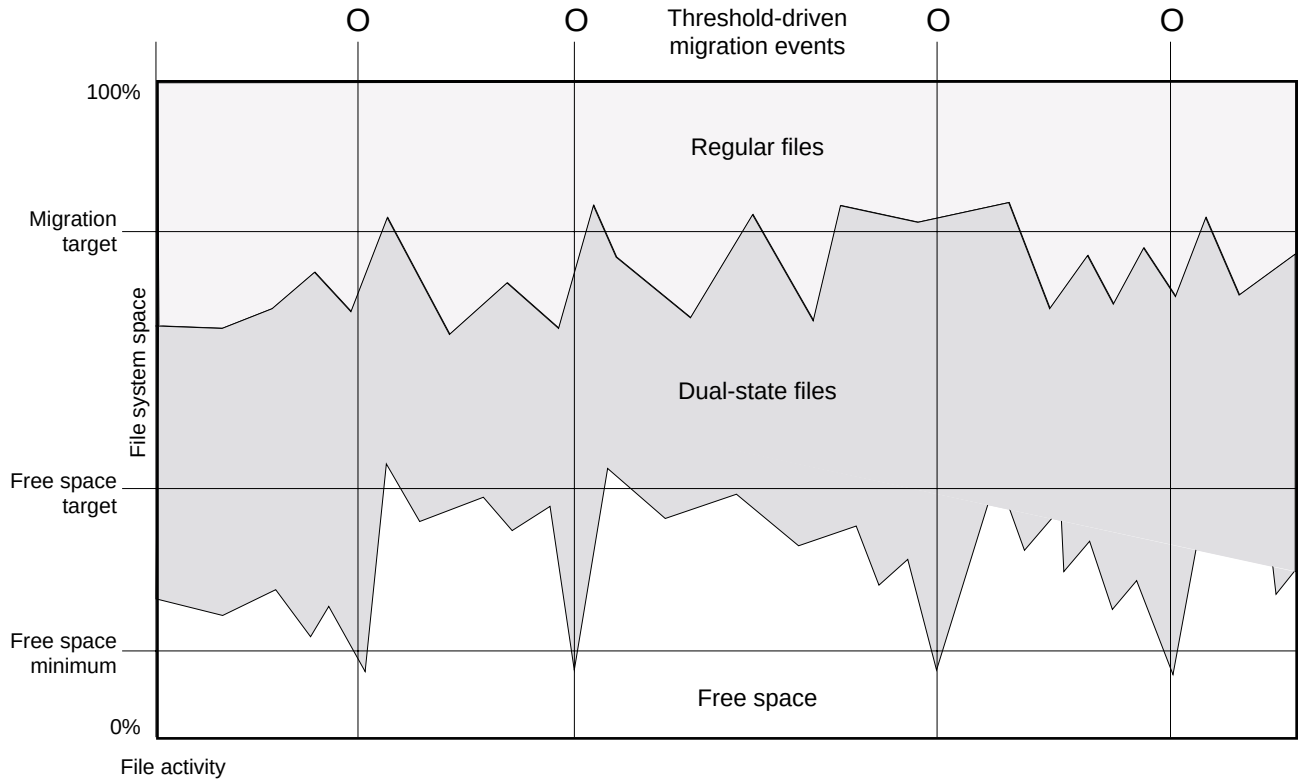

Figure 11-1 Relationship of Automated Space Management Targets

If `dmfsmon` does not find enough files to migrate (because all remaining files are exempt from migration), it uses another configuration parameter to decrement `FREE_SPACE_MINIMUM`.

`FREE_SPACE_DECREMENT` specifies the percentage of filesystem space by which `dmfsmon` will decrement `FREE_SPACE_MINIMUM` if it cannot find enough files to migrate to reach `FREE_SPACE_MINIMUM`. For example, suppose `FREE_SPACE_MINIMUM` is set to 10 and `FREE_SPACE_DECREMENT` is set to 2. If `dmfsmon` cannot find enough files to migrate to reach 10% free space, it will decrement `FREE_SPACE_MINIMUM` to 8 and try to find enough files to migrate so that 8% of the filesystem is free. If `dmfsmon` cannot achieve this percentage, it will decrement `FREE_SPACE_MINIMUM` to 6. `dmfsmon` will continue until it reaches a value for `FREE_SPACE_MINIMUM` that it can achieve, and it will try to maintain that

new value. dmfsmon restores FREE_SPACE_MINIMUM to its configured value when it can be achieved. The default value for FREE_SPACE_DECREMENT is 2.

Note: DMF manages real-time partitions differently than files in a normal partition. The dmfsfree command can only migrate files in the non-real-time partition; it ignores files in the real-time partition. Any configuration parameters you set will apply only to the non-real-time partition. Files in the real-time partition can be manually migrated with the commands dmget(1), dmput(1), and dmmigrate(8). Files are retrieved automatically when they are read.

Space Management and the DCM MSP

DMF prevents the disk cache manager (DCM) media-specific process (MSP) cache from filling by following the same general approach it takes with managed filesystems, with the following differences:

- The disk MSP (dmdskmsp) monitors the cache, instead of a separate monitoring program such as dmfsmon.
- The dmdskfree utility controls the movement of cache files to tape. This is analogous to dmfsfree.

Note: The DCM MSP uses parameters that are similar to those used for the disk MSP, although some names are different. See "policy Object" on page 364.

Automated Space Management Log File

All of the space-management commands record their activities in a common log, autolog.*yyyymmdd* (where *yyyymmdd* is the year, month, and day of file creation). The first space-management command to execute on a given day creates the log file for that day. This file resides in the directory *SPOOL_DIR*/*daemon_name* (The *SPOOL_DIR* value is specified by the SPOOL_DIR configuration parameter; see "base Object" on page 277). The space-management commands create the *daemon_name* subdirectory in *SPOOL_DIR* if it does not already exist. The full pathname of the common log file follows:

SPOOL_DIR/*daemon_name*/autolog.*yyyymmdd*

Each line in the `autolog` file begins with the time of message issue, followed by the name of the host where the message issuer ran, and the process number and program name of the message issuer. The remainder of the line contains informative or diagnostic information such as the following:

- Name of the filesystem being processed
- Number of files selected for migration and freeing
- Number of disk blocks that were migrated and freed
- Names of any other DMF commands executed
- Command's success or failure in meeting the migration and free-space targets

The following excerpt shows the format of an `autolog` file (line breaks shown here for readability):

```
23:39:35:702-V   zap  237082-dmfsmon /dmfusr1 - free_space=39.79, minimum=38
23:39:35:702-V   zap  237082-dmfsmon /dmfusr3 - free_space=15.48,minimum=15
23:40:55:723-I   zap  237082-dmfsmon Started 3409 for execution on /dmfusr3
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of blocks in the filesystem = 122232448
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of blocks in the free space target = 24446490 (20%)
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of blocks currently free = 18287168 (15.0%)
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of blocks to free = 6159322 (5.0%)
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of blocks in the migration target = 97785960 (80%)
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of blocks currently migrated = 74419040 (60.9%)
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of blocks to migrate = 5079752 (4.2%)
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Summary of files: online = 3760, offline = 6537, unmigrating
   = 30, partial = 0
23:40:56:782-I   zap    3409-dmfsfree /dmfusr3 - Number of candidates = 3629, rejected files = 0, rejected
   ranges = 0
23:41:31:150-I   zap    3409-dmfsfree /dmfusr3 - Migrated 5104824 blocks in 169 files
23:41:31:150-I   zap    3409-dmfsfree /dmfusr3 - Freed 6164480 blocks in 303 files
23:41:31:150-O   zap    3409-dmfsfree /dmfusr3 - Exiting: minimum reached - targets met by outstanding requests.
```

The DMF Daemon

The DMF daemon, `dmfdaemon`(8), is the core component of DMF. The daemon exchanges messages with commands, the kernel, the media-specific processes (MSPs), and the library servers (LSs).

When DMF is started, the daemon database is automatically initialized. To start the daemon manually, use the DMF startup script, as follows:

```
# service dmf start
```

Caution: For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage*.

Typically, DMF should be initialized as part of the normal system startup procedure by using a direct call in a system startup script in the `/etc/rc2.d` directory.

The following sections provide additional information:

- "Daemon Processing" on page 515
- "Daemon Database and `dmdadm`" on page 517
- "Daemon Logs and Journals" on page 525

Daemon Processing

After initialization, `dmfdaemon` performs the following steps:

1. Isolates itself as a daemon process.
2. Checks for the existence of other `dmfdaemon` processes. If another `dmfdaemon` exists, the newer one terminates immediately.
3. Initializes the daemon log.
4. Opens the daemon database.
5. Initializes the daemon request socket.
6. Initiates the MSPs and LSs.

7. Enters its main request processing.

The daemon uses log files and journal files as described in "Daemon Logs and Journals" on page 525.

The main request processing section of the DMF daemon consists of the following sequence:

1. The `select`(2) system call, which is used to wait for requests or for a default time-out interval
2. A request dispatch switch to read and process requests detected by the `select` call
3. A time processor, which checks activities (such as displaying statistics and running the administrator tasks) done on a time-interval basis

This processing sequence is repeated until a stop request is received from the `dmdstop`(8) command, which is normally executed via an `/etc/init.d` script by the `service dmf stop` command line. When a normal termination is received, the MSPs and LSs are terminated, the daemon database is closed, and the logs are completed.

A typical request to the daemon starts with communication from the requester. The requester is either the kernel (over the DMF device interface) or a user-level request (from the command pipe). A user-level command can originate from the automated space-management commands or from an individual user.

After receipt, the command is dispatched to the appropriate command processor within the daemon. Usually, this processor must communicate with an MSP or LS before completing the specified request. The commands are queued within the daemon and are also queued to a specific group of daemon database entries. All entries referring to the same file share the same BFID. The command is dormant until the reply from the MSP/LS is received or the MSP/LS terminates. When command processing is completed, a final reply is sent to the issuing process, if it still exists.

A final reply usually indicates that the command has completed or an error has occurred. Often, error responses require that you analyze the daemon log file to obtain a full explanation of the error. An error response issued immediately usually results from an invalid or incorrect request (for example, a request to migrate a file that has no data blocks). A delayed error response usually indicates a database, daemon, MSP, or LS problem.

Daemon Database and dmdadm

This section discusses the following:

- "Overview of the Daemon Database and dmdadm" on page 517
- "dmdadm Directives" on page 518
- "dmdadm Field and Format Keywords" on page 520
- "dmdadm Text Field Order" on page 524

Overview of the Daemon Database and dmdadm

The daemon database resides in the directory *HOME_DIR*/*daemon_name*. The daemon database contains information about the offline copies of a given file, as well as some information about the original file. The daemon database also contains the bit-file identifier (BFID), which is assigned when the file is first migrated.

Other information maintained on a per-entry basis includes the following:

- File size (in bytes)
- MSP or volume group (VG) name and recall path
- Date and time information, including the following:
  - Time at which the record was created
  - Time at which the record was last updated
  - A check time for use by the administrator
  - A soft-delete time, indicating when the entry was soft-deleted
- Original device and inode number
- Base portion of the original filename, if known

The dmdadm(8) command provides maintenance services for the daemon database.

dmdadm executes directives from stdin or from the command line when you use the -c option. All directives start with a directive name followed by one or more parameters. Parameters may be positional or keyword-value pairs, depending on the command. White space separates the directive name, keywords, and values.

When you are inside the dmdadm interface, you see the following prompt:

adm *command_number* >

At this point, the command has a 30–minute timeout associated with it. If you do not enter a response within 30 minutes of the prompt having been displayed, the dmdadm session terminates with a descriptive message. This behavior on all the database administrative commands limits the amount of time that an administrator can lock the daemon and MSP/LS databases from updates.

dmdadm Directives

The dmdadm directives are as follows:

Directive	Description
count	Displays the number of records that match the expression provided.
create	Creates a record.
delete	Deletes the specified records.
dump	Prints the specified records to standard out in ASCII; each field is separated by the pipe character (|).
help	Displays help.
list	Shows the fields of selected records. You may specify which fields are shown.
load	Applies records to the daemon database obtained from running the dump directive.
quit	Stops program execution after flushing any changed database records to disk. The abbreviation q and the string exit produce the same effect.
set	Specifies the fields to be shown in subsequent list directives.
update	Modifies the specified records.

The syntax for the dmdadm directives is as follows:

count *selection* [*limit*]
create *bfid settings*
delete *selection* [*limit*]
dump *selection* [*limit*]
help

`list` *selection* [*limit*] [*format*]
`load` *filename*
`quit (or q or exit)`
`set` *format*
`update` *selection* [*limit*] `to` *settings*`...`

where:

- The *selection* parameter specifies the records to be acted upon.
- The *limit* parameter restricts the records acted upon.
- The *bfid* parameter for the `create` directive specifies the bit-file identifier (BFID) for the record being created.
- The *settings* parameter for the `create` and `update` directives specifies one or more fields and their values.
- The *format* parameter selects the way in which output is displayed. Any program or script that parses the output from this command should explicitly specify a format; otherwise the default is used, which may change from release to release.

The value for *selection* can be one of the following:

- A BFID or range of BFIDs
- The keyword `all`
- A period (`.`), which recalls the previous selection
- An expression involving any of the above, field value comparisons, `and`, `or`, or parentheses

A field value comparison may use the following to compare a field keyword to an appropriate value:

< (less than)
> (greater than)
= (equal to)
!= (not equal to)
<= (less than or equal to)
>= (greater than or equal to)

The syntax for *selection* is as follows:

```
selection       ::=   or-expr
or-expr         ::=   and-expr [ or or-expr ]
and-expr        ::=   nested-expr [ and or-expr ]
nested-expr     ::=   comparison | ( or-expr )
comparison      ::=   bfid-range | field-keyword op field-value
op              ::=   < | > | = | != | >= | <=
bfid-range      ::=   bfid [ - bfid] | [bfid - [bfid]] | key-macro
key-macro       ::=   all
field-keyword   ::=   name or abbreviation of the record field
field-value     ::=   appropriate value for the field
bfid            ::=   character representation of the bfid
```

Thus valid values for *selection* could be any of the following:

```
305c74b200000010-305c74b200000029
7fffffff000f4411-
-305c74b2000004c8
all
origsize>1m
. and origage<7d
```

dmdadm Field and Format Keywords

The *field* parameter keywords listed below can be used as follows:

- In a *selection* parameter to select records
- In a *settings* parameter as part of a keyword-value pair, in order to specify new values for a field
- In a *format* parameter

When specifying new values for fields, some of the field keywords are valid only if you also specify the `-u` (unsafe) option.

Keyword	Description
`checkage` (`ca`)	The time at which the record was last checked; the same as `checktime`, except that it is specified as an *age string* (see below). Valid only in unsafe (`-u`) mode.

`checktime` (`ct`)	The time at which the record was last checked; an integer that reflects raw UNIX or Linux time. Valid only in unsafe (`-u`) mode.
`deleteage` (`da`)	The time at which the record was soft-deleted; the same as `deletetime`, except that it is specified as an age string. Valid only in unsafe (`-u`) mode.
`deletetime` (`dt`)	The time at which the record was soft-deleted; an integer that reflects raw UNIX or Linux time. Valid only in unsafe (`-u`) mode.
`mspname` (`mn`)	The name of the MSP or VG with which the file is associated; a string of up to 8 characters. Valid only in unsafe (`-u`) mode.
`mspkey` (`mk`)	The string that the MSP or VG can use to recall a record; a string of up to 50 characters. Valid only in unsafe (`-u`) mode.
`origage` (`oa`)	Time at which the record was created; the same as `origtime`, except that it is specified as an age string.
`origdevice` (`od`)	Original device number of the file; an integer.
`originode` (`oi`)	Original inode number of the file; an integer.
`origname` (`on`)	Base portion of the original filename; a string of up to 14 characters.
`origsize` (`os`)	Original size of the file; an integer.
`origtime` (`ot`)	Time at which the record was created; an integer that reflects raw UNIX or Linux time.
`origuid` (`ou`)	Original user ID of the record; an integer.
`updateage` (`ua`)	Time at which the record was last updated; the same as `updatetime`, except that it is specified as an age string.
`updatetime` (`ut`)	Time at which the record was last updated; an integer that reflects raw UNIX or Linux time.

The time field keywords (`checktime`, `deletetime`, `origtime`, and `updatetime`) can have one of the following values:

- `now`

- UNIX or Linux *raw time* (that is, seconds since January 1, 1970)

These keywords display their value as raw time. The value comparison > used with the date keywords means newer than the value given. For example, `>36000` is newer than 10AM on January 1, 1970, and `>852081200` is newer than 10AM on January 1, 1997.

The age field keywords (`checkage`, `deleteage`, `origage`, and `updateage`) let you express time as a string. They display their value as an integer followed by the following:

`w` (weeks)
`d` (days)
`h` (hours)
`m` (minutes)
`s` (seconds)

For example, `8w12d7h16m20s` means 8 weeks, 12 days, 7 hours, 16 minutes, and 20 seconds old.

The comparison > used with the age keywords means older than the value given (that is, `>5d` is older than 5 days).

A *limit* parameter in a directive restricts the records acted upon. It consists of one of the following keywords followed by white space and then a value:

Keyword	**Description**
`recordlimit` (`rl`)	Limits the number of records acted upon to the value that you specify; an integer.
`recordorder` (`ro`)	Specifies the order that records are scanned: • `bfid`, which specifies that the records are scanned in BFID order. • `data`, which specifies that the records are scanned in the order in which they are found in the daemon database data file. `data` is more efficient for large databases, although it is essentially unordered.

The *format* parameter selects a format to use for the display. If, for example, you want to display fields in a different order than the default or want to include fields that are not included in the default display, you specify them with the format parameter. The format parameter in a directive consists of one of the following:

`format default`
`format keyword`
`format` *field-keywords*

The `format keyword` form is intended for parsing by a program or script and therefore suppresses the headings.

The *field-keywords* may be delimited by colons or white space; white space requires the use of quotation marks.

Note: `BFID` is always included as the first field and need not be specified.

For any field that takes a byte count, you may append one of the following letters (in either uppercase or lowercase) to the integer to indicate that the value is to be multiplied (all of which are powers of 1000, not 1024):

`k` or `K` for 1 thousand
`m` or `M` for 1 million
`g` or `G` for 1 billion

The following is sample output from the `dmdadm list` directive; `recordlimit 20` specifies that you want to see only the first 20 records.

```
adm 3>list all recordlimit 20
            BFID     ORIG        ORIG   ORIG MSP      MSP
                      UID        SIZE    AGE NAME     KEY
-----------------------------------------------------------------------
305c74b200000010    20934    69140480   537d silo1    88b49f
305c74b200000013    26444      279290   537d silo1    88b4a2
305c74b200000014    10634       67000   537d silo1    88b4a3
305c74b200000016    10634   284356608   537d silo1    88b4a5
305c74b200000018    10634     1986560   537d silo1    88b4a7
305c74b20000001b    26444      232681   537d silo1    88b4aa
305c74b20000001c    10015     7533688   537d silo1    88b4ab
305c74b200000022    8964     23194990   537d silo1    88b4b1
305c74b200000023    1294    133562368   537d silo1    88b4b2
305c74b200000024    10634       67000   537d silo1    88b4b3
305c74b200000025    10634   284356608   537d silo1    88b4b4
305c74b200000026    10634     1986560   537d silo1    88b4b5
305c74b200000027    1294      1114112   537d silo1    88b4b6
305c74b200000028    10634       25270   537d silo1    88b4b7
305c74b200000029    1294     65077248   537d silo1    88b4b8
```

```
305c74b20000002b    9244      2740120   537d silo1    88b4ba
305c74b200000064    9335         9272   537d silo1    88b4f3
305c74b200000065    9335        10154   537d silo1    88b4f4
305c74b200000066    9335         4624   537d silo1    88b4f5
305c74b200000067    9335        10155   537d silo1    88b4f6
adm 4>
```

The following example displays the number of records in the daemon database that are associated with user ID 11789 and that were updated during the last five days:

```
adm 3>count origuid=11789 and updateage<5d
72 records found.
```

dmdadm Text Field Order

The text field order for daemon records generated by the dmdump(8), dmdumpj(8), and the dump directive in dmdadm is listed below. This is the format expected by the load directives in dmdadm:

1. bfid
2. origdevice
3. originode
4. origsize
5. origtime
6. updatetime
7. checktime
8. deletetime
9. origuid
10. origname
11. mspname
12. mspkey

To isolate the mspname and mspkey from the daemon records soft-deleted fewer than three days ago, use the following command:

```
dmdadm -c "dump deleteage<3d and deletetime>0" | awk "-F|" '(print $11,$12}'
```

Daemon Logs and Journals

The DMF daemon uses log files to track various types of activity. Journal files are used to track daemon database transactions.

The ASCII log of daemon actions has the following format (*SPOOL_DIR* refers to the directory specified by the SPOOL_DIR configuration parameter):

SPOOL_DIR/*daemon_name*/dmdlog.*yyyymmdd*

The convention is that *yyyy*, *mm*, and *dd* correspond to the date on which the log file was created (representing year, month, and day, respectively). Log files are created automatically by the DMF daemon.

Note: Because the DMF daemon will continue to create log files and journal files without limit, you must remove obsolete files periodically by configuring the run_remove_logs and run_remove_journals tasks in the configuration file, as described in "taskgroup Object" on page 304.

The DMF daemon automatically creates journal files that track daemon database transactions. They have the following pathname format (*JOURNAL_DIR* refers to the directory defined by the JOURNAL_DIR configuration parameter):

JOURNAL_DIR/*daemon_name*/dmd_db.*yyyymmdd[.hhmmss]*

Existing journal files are closed and new ones created in two circumstances:

- When the first transaction after midnight occurs
- When the journal file reaches size defined by the JOURNAL_SIZE configuration parameter

When the first transaction after midnight occurs, the existing open journal file is closed, and the suffix .235959 is appended to the current filename no matter what the time (or date) of closing. The closed file represents the last (or only) transaction log of the date *yyyymmdd*. A new journal file with the current date is then created.

When the journal file reaches JOURNAL_SIZE, the file is closed and the suffix *.hhmmss* is added to the name; *hh*, *mm*, and *ss* represent the hour, minute, and second of file closing. A new journal file with the same date but no time is then created.

For example, the following shows the contents of a *JOURNAL_DIR/daemon_name* directory on 15 June 1998:

```
dmd_db.19980604.235959  dmd_db.19980612.235959
dmd_db.19980605.235959  dmd_db.19980613.145514
dmd_db.19980608.235959  dmd_db.19980613.214233
dmd_db.19980609.235959  dmd_db.19980613.235959
dmd_db.19980610.235959  dmd_db.19980614.235959
dmd_db.19980611.094745  dmd_db.19980615
dmd_db.19980611.101937
dmd_db.19980611.110429
dmd_db.19980611.235959
```

For every date on which daemon database transactions occurred, there will exist a file with that date and the suffix .235959, with the exception of an existing open journal file. Some dates have additional files because the transaction log reached JOURNAL_SIZE at a specified time and the file was closed.

You can configure daemon_tasks parameters to remove old journal files (using the run_remove_journals.sh task and the JOURNAL_RETENTION parameter. For more information, see "taskgroup Object" on page 304.

Warning: If a daemon database becomes corrupt, recovery consists of applying journals to a backup copy of the database. Database recovery procedures are described in "Recovery and Restoration of DMF Database" on page 646.

The DMF Lock Manager

The dmlockmgr(8) process must be executing at all times for any DMF process to safely access and update a DMF database. The dmlockmgr process and its clients — such as dmatls, dmfdaemon(8), dmvoladm(8), and dmcatadm(8) — communicate through files, semaphores, and message queues. There are times when abnormal process terminations will result in non-orderly exit processing that will leave files and/or interprocess communication (IPC) resources allocated. As a DMF administrator, periodically you will want to look for these resources to remove them.

Note: *HOME_DIR* and *SPOOL_DIR* refer to the values of the HOME_DIR and SPOOL_DIR parameter, respectively, in the DMF configuration file. See "base Object" on page 277.

The dmlockmgr files used by the database utilities are found in several different places. There are the following types of files:

- "dmlockmgr Communication and Log Files" on page 527
- "dmlockmgr Individual Transaction Log Files" on page 529

dmlockmgr Communication and Log Files

The dmlockmgr communication and activity log files are all found in a directory formed by *HOME_DIR*/RDM_LM. The *HOME_DIR*/RDM_LM and *HOME_DIR*/RDM_LM/ftok_files directories contain the token files used to form the keys that are used to create and access the IPC resources necessary for the dmlockmgr to communicate with its clients, its standard output file, and the transaction file.

The dmlockmgr token files have the form shown in Table 13-1 on page 528.

Table 13-1 `dmlockmgr` Token Files

File	Description
HOME_DIR/RDM_LM/dmlockmgr	Used by the `dmlockmgr` and its clients to access `dmlockmgr`'s semaphore and input message queue
HOME_DIR/RDM_LM/ftok_files/ft*nnnn*	Preallocated token files that are not currently in use. As processes attempt to connect to `dmlockmgr`, these files will be used and renamed as described below. *nnnn* is a four-digit number 0000-0099.
HOME_DIR/RDM_LM/ftok_files/ft*nnnn.xxxpid*	The renamed version of the preallocated token files. *nnnn* is a four-digit number 0000-0099. *xxx* is a three-character process identifier with the following meaning: • `atr` = `dmatread` • `ats` = `dmatsnf` • `cat` = `dmcatadm` • `ddb` = `dmdadm` • `dmd` = `dmfdaemon` • `dmv` = `dmmove` • `hde` = `dmhdelete` • `lfs` = `dmloadfs` • `lib` = `dmatls` • `sel` = `dmselect` • `vol` = `dmvoladm` *pid* is the numeric process ID of the process connected to `dmlockmgr`.

The IPC resources used by DMF are always released during normal process exit cleanup. If one of the `dmlockmgr` client processes dies without removing its message queue, `dmlockmgr` will remove that queue when it detects the death of the client. The token files themselves are periodically cleaned up by the `dmlockmgr` process.

Note: Normally, the dmlockmgr process is terminated as part of normal shutdown procedures. However if you wish to stop dmlockmgr manually, you must use the following command:

```
/usr/sbin/dmclripc -u dmlockmgr -z HOME_DIR/RDM_LM
```

This command will do all of the necessary IPC resource and token file maintenance.

If the dmlockmgr process aborts, all DMF processes must be stopped and restarted in order to relogin to a new dmlockmgr process. If the dmfdaemon or dmatls processes abort during a period when the dmlockmgr has died, when they restart they will attempt to restart the dmlockmgr. The new dmlockmgr process will detect existing DMF processes that were communicating with the now-dead copy of dmlockmgr, and it will send a termination message to those DMF processes.

The dmlockmgr maintains a log file that is named as follows, where *yyyy*, *mm*, and *dd* are the year, month, and day:

HOME_DIR/RDM_LM/dmlocklog.*yyyymmdd*

The log file is closed and a new one opened at the first log request of a new day, although these files typically are not large. These log files are removed via the run_remove_log.sh daemon task command. For more information about run_remove_log.sh, see "taskgroup Object" on page 304.

dmlockmgr Individual Transaction Log Files

The individual transaction log files have the following form:

prefix.log

where *prefix* is the same format as the token filename described in Table 13-1 on page 528 as ft*nnnn.xxxpid*. The prefix associates a log file directly with the token file of the same name.

Most of these log files will be created in the *HOME_DIR* under the daemon's and library servers' subdirectories. In almost all cases, the processes that create these log files will remove them when they exit. However, if a process terminates abnormally, its log file may not be removed. Transaction log files can sometimes become quite large, on the order of 10's of Mbytes. Most of these orphaned log files will be removed by the daemon as part of its normal operation.

Several DMF commands allow accessing copies of database files in places other than the *HOME_DIR*. If an orphaned log is encountered in a location other than in the *HOME_DIR*, it may be removed after it is clear that it is no longer in use. In order to verify that it is no longer in use, search the *HOME_DIR*/RDM_LM/ftok_files directory for a file with the same name as the prefix of the log file. If no such ftok_files file exists, it is safe to remove the log file.

The transaction activity file, *HOME_DIR*/RDM_LM/vista.taf, is the transaction log file that contains information about active transactions in the system. It is used to facilitate automatic database transaction processing.

Caution: Do **not** delete the *HOME_DIR*/RDM_LM/vista.taf file.

Library Servers and Media-Specific Processes

This chapter discusses the following for library servers (LSs) and media-specific processes (MSPs):

- "LS Operations" on page 532
- "Cloud MSP" on page 569
- "FTP MSP" on page 573
- "Disk MSP" on page 576
- "DCM MSP" on page 579
- "`dmdskvfy` Command" on page 579
- "Moving Migrated Data" on page 579
- "LS Error Analysis and Avoidance" on page 580
- "LS Drive Scheduling" on page 582
- "LS Status Monitoring" on page 582

LS Operations

This section discusses the following:

- "LS Overview" on page 532
- "LS Directories" on page 533
- "Media Concepts" on page 534
- "CAT Records" on page 536
- "VOL Records" on page 537
- "LS Journals" on page 538
- "LS Logs" on page 539
- "Volume Merging" on page 543
- "`dmcatadm` Command" on page 544
- "`dmvoladm` Command" on page 554
- "`dmatread` Command" on page 567
- "`dmatsnf` Command" on page 568
- "`dmaudit verifymsp` Command" on page 568

LS Overview

The LS consists of the following programs:

```
dmatls
dmatwc
dmatrc
```

The DMF daemon executes `dmatls` as a child process. In turn, `dmatls` executes `dmatwc` (the write child) to write data to tape and `dmatrc` (the read child) to read data from tape.

The `dmatls` program maintains the following records in the LS database:

- Catalog (CAT) records, which contain information about the files that the LS maintains

- Volume (VOL) records, which contain information about the media that the LS uses

The database files are not text files and cannot be updated by standard utility programs. Detailed information about the database files and their associated utilities is provided in "CAT Records" on page 536 and "VOL Records" on page 537.

The LS provides a mechanism for copying active data from volumes that contain largely obsolete data to volumes that contain mostly active data. This process is referred to as *volume merging*. Data on LS volumes becomes obsolete when users delete or modify their files. Volume merging can be configured to occur automatically (see "LS Tasks" on page 441). It can also be triggered by marking LS volumes as sparse with the `dmvoladm`(8) command.

The LS provides the following utilities that read LS volumes directly:

- `dmatread`(8) copies all or part of a migrated file to disk
- `dmatsnf`(8) audits and verifies LS volumes

LS Directories

Each instance of the LS needs three types of directories, one for each of the following:

- Database files for CAT and VOL records
- Database journal files
- Log files

Sites define the location of these directories by editing the base object configuration file parameters `HOME_DIR`, `JOURNAL_DIR`, and `SPOOL_DIR`, whose values are referred to as *HOME_DIR*, *JOURNAL_DIR*, and *SPOOL_DIR* in this document. A given instance of the LS creates a subdirectory named after itself in each of these three directories.

For example, if an instance of the LS is called `cart1`, its database files reside in directory *HOME_DIR*`/cart1`. If another instance of the LS is called `cart2`, its database files reside in *HOME_DIR*`/cart2`. If an instance of the LS is called `cart3`, its database files reside in *HOME_DIR*`/cart3`.

Similarly, LS `cart1` stores its journal files in directory *JOURNAL_DIR*`/cart1` and its log files and other working files in *SPOOL_DIR*`/cart1`.

Media Concepts

The LS takes full advantage of the capabilities of modern media devices, including data compression and fast media positioning. To accommodate these capabilities and to provide recovery from surface or other media defects, `dmatls` uses a number of structural concepts built on top of traditional media structure.

The components are as follows:

- The *block* is the basic structural component of most media technologies. It is the physical unit of I/O to and from the media. The optimal block size varies with the device type. For example, the default block size for an STK T10000A tape drive is 524288 bytes.
- A *chunk* is as much or as little of a user file as fits on the remainder of the media (see Figure 14-1 on page 536). Thus, every migrated file has at least one, and sometimes many, chunks. Such a concept is necessary because the capacity of a volume is unknown until written, both because of natural variation in the medium itself and because the effect of data compression varies with the data contents.
- A *zone* is a logical block containing many physical blocks ending with a media mark. A zone has a target size that is configurable by media type. The default zone size is 50000000 bytes.

  The VG writes chunks into the zone until one of three conditions occurs:

  – The zone size is exceeded

  – The VG exhausts chunks to write

  – The end of media is encountered

  Thus, the actual zone size can vary from well below the target size to the entire volume. A zone never spans physical volumes.

  The zone plays several roles:

  – The zone size is the amount of data that triggers `dmatls` to start a process to write files to secondary storage.

  – The LS maintains the beginning of each zone in its database. This allows the LS to use fast hardware positioning functions to return to the beginning, so that it can restore the chunks in that zone.

Because getting the media position and writing a media mark can be very costly, the concept of a zone and the target size provides a way to control the trade offs between write performance, safety, and recall speed.

Figure 14-1 illustrates the way files are distributed over chunks, zones, and volumes, depending upon the file size. In this example, the tape with volume serial number (VSN) `VOL001` has two zones and contains six files and part of a seventh. The tapes with VSNs `VOL002` and `VOL003` contain the rest of file `g`. Notice that on `VOL001` file `g` is associated with chunk 7, while on the other two tapes it is associated with chunk 1. File `g` has three VSNs associated with it, and each tape associates the file with a chunk and zone unique to that tape.

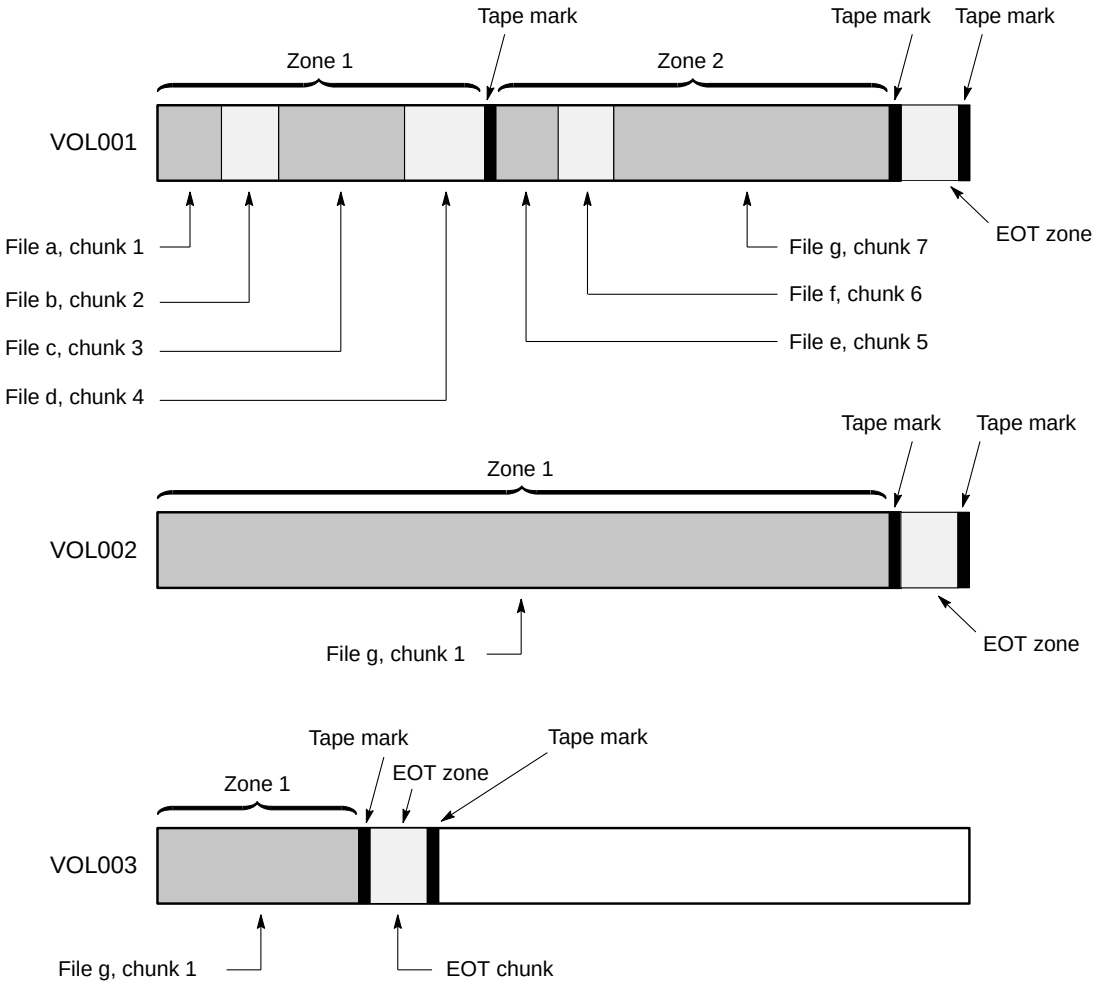

Figure 14-1 Media Concepts

CAT Records

Catalog (CAT) records store the location of each file chunk in terms of its volume, zone, and chunk number. The key for these records is the file's bit-file identifier (BFID).

Note: You do not explicitly create CAT records; they are created when files migrate.

There are the following files:

CAT Files	Description
`tpcrdm.dat`	Contains the catalog data records
`tpcrdm.key1.keys`, `tpcrdm.key2.keys`	Contains the indexes to the catalog data

The `libsrv_db.dbd` LS database definition file in the same directory describes the CAT record files and their record structure.

All files are non-ASCII and cannot be maintained by standard utility programs. The `dmcatadm` command provides facilities to create, query, and modify CAT records (see "`dmcatadm` Command" on page 544).

Note: The ability to create or modify CAT records with `dmcatadm` is provided primarily for testing or error recovery purposes. In the normal course of operations, you would never use this capability.

VOL Records

Volume (VOL) records in the LS database contain information about each volume that exists in the pool of tapes to be used by `dmatls`. These records are indexed by the volume serial number (VSN) of each volume and contain information such as the following:

- Volume's type
- Estimated capacity
- Label type
- A number of flags indicating the state of the volume
- VG or allocation group

Note: Unlike CAT records, you must create VOL records before using `dmatls` for the first time.

There are the following files:

VOL Files	Description
`tpvrdm.dat`	Contains the volume data records
`tpvrdm.vsn.keys`	Contains the indexes to the volume data records

The `libsrv_db.dbd` LS database definition file in the same directory describes the VOL record files and their record structure.

The files contain binary data and require special maintenance utilities. The `dmvoladm` command provides facilities to create, query, and modify VOL records; for more information, see "`dmvoladm` Command" on page 554. Additional database maintenance utilities are described in "Recovery and Restoration of DMF Database" on page 646.

Note: If you have more than one instance of a VG, you must ensure that the volume sets for each are mutually exclusive.

LS Journals

Each instance of `dmatls` protects its database by recording every transaction in a journal file. Journal file pathnames have the following format:

JOURNAL_DIR`/`*ls_name*`/libsrv_db.`*yyyymmdd[.hhmmss]*

The LS creates journal files automatically.

Existing journal files are closed and new ones created in two circumstances:

- When the first transaction after midnight occurs
- When the journal file reaches the size defined by the `JOURNAL_SIZE` configuration parameter

When the first transaction after midnight occurs, the existing open journal file is closed and the suffix `.235959` is appended to the current filename no matter what

the time (or date) of closing. The closed file represents the last (or only) transaction log of the date *yyyymmdd*. A new journal file with the current date is then created.

When the journal file reaches JOURNAL_SIZE, the file is closed and the suffix *.hhmmss* is added to the name; *hh*, *mm*, and *ss* represent the hour, minute, and second of file closing. A new journal file with the same date but no time is then created.

For example, the following shows the contents of a *JOURNAL_DIR/ls_name* directory on 15 June 2004:

```
libsrv_db.20040527.235959  libsrv_db.20040606.235959
libsrv_db.20040528.235959  libsrv_db.20040607.235959
libsrv_db.20040529.235959  libsrv_db.20040608.235959
libsrv_db.20040530.235959  libsrv_db.20040609.235959
libsrv_db.20040531.235959  libsrv_db.20040610.235959
libsrv_db.20040601.235959  libsrv_db.20040611.235959
libsrv_db.20040602.235959  libsrv_db.20040612.235959
libsrv_db.20040603.235959  libsrv_db.20040613.235959
libsrv_db.20040604.235959  libsrv_db.20040614.235959
libsrv_db.20040605.235959  libsrv_db.20040615
```

For every date on which LS database transactions occurred, there will exist a file with that date and the suffix .235959, with the exception of an existing open journal file. Some dates may have additional files because the transaction log reached JOURNAL_SIZE at a specified time and the file was closed.

You can configure daemon_tasks parameters to remove old journal files (using the run_remove_journals.sh task and the JOURNAL_RETENTION parameter. For more information, see "taskgroup Object" on page 304.

If an LS database becomes corrupt, recovery consists of applying the journal files to a backup copy of the database.

LS Logs

All DMF MSPs and LSs maintain log files named msplog.*yyyymmdd* in the MSP/LS spool directory which, by default, is *SPOOL_DIR/mspname*. *SPOOL_DIR* is configured in the base object of the configuration file; *mspname* is the name of the MSP/LS in the daemon object of the configuration file; *yyyymmdd* is the current year, month, and day.

These log files are distinct from the logs maintained by the DMF daemon; however, some of the messages that occur in the daemon log are responses that the MSP/LS generates. The content of the log is controlled by the MESSAGE_LEVEL configuration

parameter. For a description of the levels of logging available, see Table 10-1 on page 508 and the `dmf.conf`(5) man page.

The `msplog.`*yyyymmdd* file is the primary log for the LS and contains most of the messages. This file is written by `dmatls`. In addition, `dmatrc` and `dmatwc` create a `moverlog.`*yyyymmdd* log file each day in the subdirectory `moverlogs/`*hostname*.

This section describes informational statistics provided by the tape log files. These messages appear in the *SPOOL_DIR*`/`*msp_name*`/msplog.`*yyyymmdd* files. Timing information provided (such as MB transferred per second) should not be used as an accurate benchmark of actual data transfer rates. This information is provided for monitoring DMF and should only be used in comparison to similar data provided by DMF. Text in all uppercase references a parameter defined in the DMF configuration file. For more information, see Chapter 7, "DMF Configuration File" on page 271, the comments in the sample configuration file, and the `dmf.conf`(5) man page.

Note: Because the LS will continue to create log files and journal files without limit, you must remove obsolete files periodically by configuring the `run_remove_logs.sh` and `run_remove_journals.sh` tasks in the configuration file, as described in "`taskgroup` Object" on page 304.

Example 14-1 LS Statistics Messages

The following is an example of LS statistics messages taken from an `msplog.`*yyyymmdd* file. These messages are automatically and periodically issued by the LS.

```
08:46:00:404-I      zap   237076-dmatls vg1.stats: children=2/2/0/2, btp=672617104/527956913/0, wc=1/2, cwc=?
08:46:00:404-I      zap   237076-dmatls vg2.stats: children=0/0/0/2, btp=0/0/0, wc=0/2, cwc=?
08:46:00:404-I      zap   237076-dmatls vg1.stats: data put=92957.718 mb, data recalled=24964.680 mb
08:46:00:404-I      zap   237076-dmatls vg2.stats: data put=1239.537 mb, data recalled=1120.492 mb
08:46:00:404-I      zap   237076-dmatls vg1.stats:     Put_File -      0   8900      0    282      0
08:46:00:404-I      zap   237076-dmatls vg1.stats:     Get_File -      0   1809      0      0      0
08:46:00:404-I      zap   237076-dmatls vg1.stats:  Delete_File -      0 107618      0      0      0
08:46:00:404-I      zap   237076-dmatls vg1.stats:   Cancel_Req -      0    282      0      0      0
08:46:00:404-I      zap   237076-dmatls vg1.stats:     Flushall -      0      5      0      0      0
08:46:00:404-I      zap   237076-dmatls vg1.stats:        Merge -     44      0      0      0      0
08:46:00:404-I      zap   237076-dmatls vg2.stats:     Put_File -      0   1850      0    211      0
08:46:00:404-I      zap   237076-dmatls vg2.stats:     Get_File -      0     68      0      0      0
08:46:00:404-I      zap   237076-dmatls vg2.stats:  Delete_File -      0      4      0      0      0
08:46:00:404-I      zap   237076-dmatls vg2.stats:   Cancel_Req -      0    211      0      0      0
08:46:00:404-I      zap   237076-dmatls vg2.stats:     Flushall -      0      1      1      0      0
08:46:00:404-I      zap   237076-dmatls vg1.stats: mc=2, ms=2000000000, mu=679346176, sm=0
```

The information provided by these entries is defined as follows:

- `children=2/2/0/2` represents the total child processes (`2`), the active child processes (`2`), the clean processes running (`0`), and the current maximum number of children the VG may have (`2`). Clean children are used when a `dmatrc` or `dmatwc` process dies without cleaning up.
- `btp=672617104/527956913/0` represents the bytes queued for putting (`672617104`), the threshold at which to start the next put child (`527956913`), and the bytes assigned to socket I/O (`0`)
- `wc=1/2` represents the active write child processes (`1`) and the configured value of `MAX_PUT_CHILDREN` (`2`)
- `cwc=?` represents the host name and process ID of the current write child (that is, the write child that is accepting data to write). `?` represents none.

The next set of lines gives the total amount of data put (such as `92957.718` MB) and recalled (such as `24964.680` MB).

The next set of six lines provide statistics for each type of VG request. Statistics information is provided only for requests that have been issued since the LS was started. These lines have the following format:

request_name active successful errors canceled forwards

where:

active	Represents the number of requests not yet completed.
successful	Represents the number of successfully completed requests.
error	Represents the number of requests that completed with errors.
canceled	Represents the number of canceled requests.
forwards	Represents the number of requests that were returned to the DMF daemon so it could forward them to another VG. For example, if the volume for a recall request in the primary VG was not in the library, but a copy existed in a secondary VG, the primary VG would return the request to the DMF daemon which would then forward it to the secondary VG, and the value displayed in this column would be incremented.

The last set of lines provide the following information:

- `mc` is the configured value for `MERGE_CUTOFF`, the cutoff to stop scheduling media for merging (such as `2`)
- `ms` is the configured value for `CACHE_SPACE`, the merge cache space available (such as `2000000000` bytes)
- `mu` is the merge cache space used (such as `679346176` bytes)
- `sm` is the number of socket merge children (`0`)

The LS write child (`dmatwc`) and read child (`dmatrc`) also produce statistics messages in the `moverlog` file. These messages contain timing statistics whose format changes from release to release, and they are not documented in this manual.

Volume Merging

Note: Merging is not appropriate for a volume configured as a fast-mount cache.

When users delete or modify their migrated files, the copy that is on secondary storage becomes obsolete. Over time, some volumes will become entirely empty and can be reused. However, most volumes experience a gradual increase in the ratio of obsolete data to active data; such volumes are said to be *sparsely populated* or *sparse*. To reclaim the unused space on these volumes, DMF provides a *volume merge* facility, which copies the active data from several sparse volumes to a new volume, thus freeing the sparse volumes for reuse. Volume merging can be configured to occur automatically by using the `run_merge_tapes.sh` or `run_merge_mgr.sh` tasks (see "LS Tasks" on page 441).

Volume merging can also be done manually. `dmatls` performs merge operations whenever sparse volumes and the necessary resources exist at the same time. Use the `dmvoladm select` directive to mark VG volumes as sparse. (The `select` directive is described in "`dmvoladm` Command" on page 554.) Because the merge processing occurs simultaneously with other DMF activities, it is easiest to configure DMF to automatically perform merges at night or during other periods of relatively low activity.

The `dmatls` utility can perform volume-to-volume merging. Volume-to-volume merging is accomplished by moving data across a socket connection between the LS read-child and the LS write-child. The benefit of using a socket to transfer data between volumes is that you do not have to reserve disk space. The drawback to using a socket for data transfer is the cost of linking the process that performs the read with the process that performs the write.

In busy environments that have heavy contention for drives, the close coupling between the socket's reader and writer can be costly, especially when short files are being transferred. For large files, the overhead and possible delays in waiting for both volumes to be mounted is small compared to the benefit of rapid transfer and zero impact on free disk space. For this reason, you can move small files through a disk cache and big files through a socket. This process is mediated by the following configuration parameters:

```
CACHE_DIR
CACHE_SPACE
MAX_CACHE_FILE
MERGE_CUTOFF
```

For more information, see Chapter 7, "DMF Configuration File" on page 271.

Using a small amount of disk space to hold small chunks can have a significant impact on the total time required to perform merges. The default configuration options are set to move 100% of merge data across sockets.

Note: It is important to avoid volume merging on more than one VG simultaneously if they share a device. If you initiate a merge process on more than one VG on the same device at the same time (either by entering the same time in the DMF configuration file or by triggering the process manually), both processes will compete for media transports. When a limited number of media transports are available, a deadlock can occur. If you chose not to configure DMF to perform merges automatically by configuring the `run_merge_tape.sh` or `run_merge_mgr.sh` tasks, ensure that your `cron` jobs that automatically initiate volume merging refrain from initiating a second merge process until after all previously initiated merges are complete. You can accomplish this by using the `dmvoladm` command within the `cron` job to check for volumes that have the `hsparse` flag, as shown in the following example for an LS with two VGs:

```
tapes=$(dmvoladm -m ls -c "count hsparse")
if [[ -z "$tapes" ]]; then
        # start merge on vg2
        dmvoladm -m ls -c "select hfull and threshold<=30 and vg=vg2"
fi
```

`dmcatadm` Command

The `dmcatadm`(8) command provides maintenance services for CAT records.

When you are inside the `dmcatadm` interface, you see the following prompt:

`adm` *command_number* `>`

At this point, the command has a 30–minute timeout associated with it. If you do not enter a response within 30 minutes of the prompt having been displayed, the `dmcatadm` session terminates with a descriptive message. This behavior on all the database administrative commands limits the amount of time that an administrator can lock the daemon database and the LS database from updates.

Note: Most of these facilities, especially the ability to create and modify CAT records in the LS database, are intended primarily for testing or error recovery purposes.

`dmcatadm` Directives

The `dmcatadm` command executes directives from `stdin` or from the command line when you use the `-c` option. All directives start with a directive name followed by one or more parameters. Parameters may be positional or keyword-value pairs, depending on the command. White space separates the directive name, keywords, and values.

The `dmcatadm` directives are as follows:

Directive	Description
`count`	Displays the number of records that match the expression provided.
`create`	Creates a CAT record.
`delete`	Deletes the specified CAT records.
`dump`	Prints the specified CAT records to standard out in ASCII; each field is separated by the pipe character (`|`).
`help`	Displays help.
`list`	Shows the fields of selected CAT records. You may specify which fields are shown.
`load`	Applies records to the LS database obtained from running the `dump` directive.
`quit`	Stops program execution after flushing any changed database records to disk. The abbreviation `q` and the string `exit` produce the same effect.
`set`	Specifies the fields to be displayed in subsequent `list` directives.
`update`	Modifies the specified CAT records.
`verify`	Verifies the LS database against the daemon database.

The first parameter of most directives specifies the records to manipulate, and the remaining parameters are keyword-value pairs.

The syntax for the dmcatadm directives is summarized as follows:

```
count selection [limit]
create bfid settings ...
delete selection [limit]
dump selection [limit]
help
list selection [limit] [format]
load filename
quit (or q or exit)
set [format]
update selection [limit] to settings...
verify selection [entries] [vgnames] [limit]
```

The parameters are as follows:

- The *selection* parameter specifies the records to be acted upon. The value for *selection* can be one of the following:
  - A *bfid* or range of *bfids* in the form *bfid* [-] [*bfid*]. *bfid*- specifies all records starting with *bfid*, and -*bfid* specifies all records up to *bfid*.
  - The keyword all
  - A period (.), which recalls the previous selection
  - An expression involving any of the above, field value comparisons, and, or, or parentheses

  A field value comparison may use the following to compare a field keyword to an appropriate value:

  < (less than)
  > (greater than)
  = (equal to)
  != (not equal to)
  <= (less than or equal to)
  >= (greater than or equal to)

The syntax for *selection* is as follows:

```
selection       ::=   or-expr
or-expr         ::=   and-expr [ or or-expr ]
and-expr        ::=   nested-expr [ and or-expr ]
nested-expr     ::=   comparison | ( or-expr )
comparison      ::=   key-range | field-keyword op field-value
op              ::=   < | > | = | != | <= | >=
bfid-range      ::=   bfid [ - bfid] | [bfid - [bfid]] | key-macro
key-macro       ::=   all
field-keyword   ::=   name or abbreviation of the record field
field-value     ::=   appropriate value for the field
key             ::=   character representation of the record bfid
```

Thus valid *selection*s could be any of the following:

```
305c74b200000010-305c74b200000029
7fffffff000f4411-
-305c74b2000004c8
all
chunkoffset>0
chunknumber>0 and writeage<5d
. and writeage>4d
vsn=S07638
```

- The *limit* parameter restricts the records acted upon.
- The *bfid* parameter for the `create` directive specifies the bit-file identifier (BFID) for the record being created. The value for *bfid* may be a BFID designator in the form of a hexadecimal number.
- The *settings* parameter for the `create` and `update` directives specify one or more fields and their values.
- The *format* parameter selects the way in which output is displayed. Any program or script that parses the output from this command should explicitly specify a format; otherwise the default is used, which may change from release to release.
- The *entries* parameter specifies a file of daemon database entries.
- The *vgnames* parameter specifies the names of the VGs associated with the records.

`dmcatadm` Keywords

You can use the *field* keywords listed below as part of the following:

- A *selection* parameter to select records
- A *format* parameter
- A *settings* parameter to specify new values for a field, in which case you must specify a keyword-value pair

A keyword-value pair consists of a keyword followed by white space and then a value. When specifying new values for fields, some of the keywords are valid only if you also specify the `-u` (unsafe) option. The abbreviation for each of the keywords is given in parenthesis following its name.

Keyword	Description
`cflags` (`cf`)	For future use.
`chunkdata` (`cd`)	Specifies the actual number of bytes written to tape by the VG for the chunk. In the case of sparse files, this field will be smaller than `chunklength`. This is valid only in unsafe (`-u`) mode.
`chunklength` (`cl`)	The size of the chunk in bytes; an integer. This is valid only in unsafe (`-u`) mode.
`chunknumber` (`cn`)	The ordinal of the chunk on its volume. For example, `1` if the chunk is the first chunk on the volume, `2` if it is the second, and so on. Not valid as part of a *settings* parameter in an `update` directive.
`chunkoffset` (`co`)	The byte offset within the file where the chunk begins; an integer. For example, the first chunk of a file has `chunkoffset 0`. If that first chunk is 1,000,000 bytes long, the second chunk would have `chunkoffset 1000000`. This is valid only in unsafe (`-u`) mode.
`chunkpos` (`cp`)	The block offset within the zone where the chunk begins — a hexadecimal integer. For example, the first chunk in a zone has `chunkpos 1`. A value of `0` means unknown. Valid only in unsafe (`-u`) mode.
`filesize` (`fs`)	The original file size in bytes, an integer. This is valid only in unsafe (`-u`) mode.

`readage` (`ra`)	The date and time when the chunk was last read; the same as `readdate`, except specified as *age*.
`readcount` (`rc`)	The number of times the chunk has been recalled to disk; an integer.
`readdate` (`rd`)	The date and time when the chunk was last read, an integer that reflects raw UNIX or Linux time.
`volgrp` (`vg`)	The VG name. This keyword is valid for LSs only. This keyword is not valid as part of a *settings* parameter. Changing this field in an existing record is valid only in unsafe (`-u`) mode.
`vsn` (`v`)	The volume serial numbers; a list of one or more 6-character alphanumeric volume serial numbers separated by colons (`:`). This keyword is not valid as part of a *settings* parameter in an `update` directive.
`writeage` (`wa`)	The date and time when the chunk was written; the same as `writedate`, except specified as *age*. This is valid only in unsafe (`-u`) mode.
`writedate`(`wd`)	The date and time when the chunk was written, an integer that reflects raw UNIX or Linux time. This is valid only in unsafe (`-u`) mode.
`zoneblockid` (`zb`)	Allows just the block ID portion of the `zonepos` to be displayed, returned, or changed. This is valid only in unsafe (`-u`) mode.
`zonenumber` (`zn`)	Allows just the zone number portion of the `zonepos` to be displayed, returned, or changed. This is valid only in unsafe (`-u`) mode.
`zonepos` (`zp`)	The physical address of the zone on the volume, expressed in the form *integer*/*hexadecimal-integer*, designating a zone number and block ID. A value of zero is used for *hexadecimal-integer* if no block ID is known. *integer* is the same as `zonenumber`, and *hexadecimal-integer* is the same as `zoneblockid`. This is valid only in unsafe (`-u`) mode.

The date field keywords (`readdate` and `writedate`) have one of the following values:

- `now`
- Raw UNIX or Linux time (seconds since January 1, 1970)

These keywords display their value as raw UNIX or Linux time. The value comparison `>` used with the date keywords means newer than the value given. For example, `>36000` is newer than 10AM on January 1, 1970, and `>852081200` is newer than 10AM on January 1, 1997.

The age field keywords (`readage` and `writeage`) let you express time as *age* in a string in a form. They display their value as an integer followed by the following:

`w` (weeks)
`d` (days)
`h` (hours)
`m` (minutes)
`s` (seconds)

For example, `8w12d7h16m20s` means 8 weeks, 12 days, 7 hours, 16 minutes, and 20 seconds old.

The comparison `>` used with the age keywords means older than the value given (that is, `>5d` is older than 5 days).

The *limit* parameter in a directive limits the records acted upon. It consists of one of the following keywords followed by white space and then a value:

Keyword	**Description**
`recordlimit` (`rl`)	Limits the number of records acted upon to the value that you specify; an integer.
`recordorder` (`ro`)	Specifies the order that records are scanned: • `data`specifies that records are scanned in the order in which they are stored in the LS database, which is fastest but essentially unordered • `key`specifies that records are scanned in ascending order of the chunk key

- `vsn` specifies that records are scanned in ascending order of the chunk VSN

The following keywords specify files of daemon database entries:

Keyword	Description
`entries` (`e`)	Specifies a file of daemon database entries. This keyword applies to the `verify` directive and consists of the word `entries` (or its abbreviation `e`) followed by a string.
`vgnames` (`vn`)	Specifies the names of the VGs associated with the record. This keyword applies to the `verify` directive and consists of the word `vgnames` (or its abbreviation `vn`) followed by a quoted, space-separated list of names.

The *format* parameter in a directive consists of the word `format` followed by white space and then one of the following:

- The word `default`
- The word `keyword` (suppresses the headings and is intended for parsing by a program or script)
- A list of field keywords, which may be delimited by colons or spaces (spaces require the use of quoting)

Note: The BFID is always included as the first field and need not be specified.

For any field that takes a byte count, you may append one of the following letters (in either uppercase or lowercase) to the integer to indicate that the value is to be multiplied (all of which are powers of 1000, not 1024):

`k` or `K` for 1 thousand
`m` or `M` for 1 million
`g` or `G` for 1 billion

For information about the role of the `dmcatadm`(8) command in database recovery, see "Recovery and Restoration of DMF Database" on page 646.

Example 14-2 `dmcatadm list` Directive

The following is sample output from the `dmcatadm list` directive. The file with key `3273d5420001e244` has two chunks because it spans two physical tape volumes; the first chunk contains bytes 0–24821759, and the second chunk bytes 24821760 (the `CHUNK OFFSET`) to the end of the file.

```
adm 3>list 3273d5420001e242- recordlimit 10
                          WRITE     CHUNK       CHUNK   CHUNK
                    KEY     AGE    OFFSET      LENGTH     NUM       VSN
-----------------------------------------------------------------------
       3273d5420001e242     61d         0    77863935      13    S12940
       3273d5420001e244     61d         0    24821760     168    S12936
       3273d5420001e244     61d  24821760    23543808       1    S12945
       3273d5420001e245     61d         0    51019776       2    S12945
       3273d5420001e246     61d         0    45629440      59    S12938
       3273d5420001e247     61d         0    35586048      60    S12938
       3273d5420001e248     61d         0     9568256       3    S12944
       3273d5420001e249     61d         0    14221312       4    S12944
       3273d5420001e24a     61d         0      458752       5    S12944
       3273d5420001e24b     61d         0    14155776       6    S12944
```

The following is sample output from the `dmcatadm list` directive for an LS. The file with key `3b4b28f2000000000000ae80` has 2 chunks because it was migrated to two different VGs within this LS. The output from the `dmvoladm list` directive that follows shows that VSN `000700` is assigned to the VG named `vg8a15`, and VSN `000727` is assigned to the VG named `vg8a05`.

```
# dmcatadm -m ls1
adm 1>list 3b4b28f2000000000000ae80- recordlimit 4
                          WRITE       CHUNK       CHUNK CHUNK
                     KEY    AGE      OFFSET      LENGTH   NUM VSN
-----------------------------------------------------------------
3b4b28f2000000000000ae80     1d           0     2305938   120 000700
3b4b28f2000000000000ae80     4d           0     2305938    32 000727
3b4b28f2000000000000ae82     1d           0      234277   247 003171
3b4b28f2000000000000ae82     1d           0      234277   186 003176

adm 2> quit

# dmvoladm -m ls1
adm 1>list vsn=000700
                                          DATA    EOT   EOT             WR/FR
VSN      VOLGRP LB      DATA LEFT        WRITTEN  CHUNK  ZONE   HFLAGS      AGE
-------------------------------------------------------------------------------
000700   vg8a15 al     150.280473     233.786093    123     9  ------u--     1d
adm 2>list vsn=000727
                                          DATA    EOT   EOT             WR/FR
VSN      VOLGRP LB      DATA LEFT        WRITTEN  CHUNK  ZONE   HFLAGS      AGE
-------------------------------------------------------------------------------
000727   vg8a05 al     159.107337     200.443980    102     6  ---------     1d
```

dmcatadm Text Field Order

The text field order for chunk records generated by the dmdump(8), dmdumpj(8), and the dump directive in dmcatadm is listed below. This is the format expected by the load directives in dmcatadm:

1. C (indicates the chunk record type)
2. bfid (hexadecimal digits)
3. filesize
4. writedata
5. readdate
6. readcount

7. `chunkoffset`
8. `chunklength`
9. `chunkdata`
10. `chunknumber`
11. `flags` (in octal)
12. `zoneposition` (`zonenumber` / `zoneblockid`) (in hexadecimal)
13. `vsn`
14. `chunkpos` (in hexadecimal)

`dmvoladm` Command

This section discusses the following:

- "`dmvoladm` Overview" on page 554
- "`dmvoladm` Directives" on page 555
- "`dmvoladm` Field Keywords" on page 557
- "`dmvoladm` Text Field Order" on page 563
- "`dmvoladm` Examples" on page 564

`dmvoladm` Overview

The `dmvoladm`(8) command provides maintenance services for VOL records. In addition to the creation and modification of volume records, `dmvoladm` has an important role in the recovery of VOL records from an LS database checkpoint and is the mechanism that triggers volume merge activity.

When you are inside the `dmvoladm` interface, you see the following prompt:

`adm` *command_number* `>`

At this point, the command has a 30–minute timeout associated with it. If you do not enter a response within 30 minutes of the prompt having been displayed, the `dmvoladm` session terminates with a descriptive message. This behavior on all the

database administrative commands limits the amount of time that an administrator can lock the daemon database and the LS database from updates.

`dmvoladm` Directives

The `dmvoladm` command executes directives from `stdin` or from the command line when you use the `-c` option. The syntax is the same as for `dmcatadm`: a directive name followed by parameters or paired keywords and values, all separated by white space.

Directive	Description
`count`	Displays the number of records that match the expression provided.
`create`	Creates a VOL record.
`delete`	Deletes the specified VOL records.
`dump`	Prints the specified VOL records to standard output in ASCII. Each field is separated by the pipe character (`|`).
`help`	Displays help.
`list`	Shows the fields of selected VOL records. You may specify which fields are shown.
`load`	Applies VOL records to the LS database obtained from running the `dump` directive.
`quit`	Stops program execution after flushing any changed records to disk. The abbreviation `q` and the string `exit` produce the same effect.
`repair`	Causes `dmvoladm` to adjust the usage information for specified volumes based on CAT records in the LS database. This directive is valid only in unsafe (`-u`) mode.
`select`	Marks selected volumes as being sparse. Equivalent to `update` *expression* `to hsparse on`.
`set`	Specifies the fields to be shown in subsequent `list` directives.
`update`	Modifies the specified VOL records.
`verify`	Verifies the LS database against the daemon database.

The syntax for the `dmvoladm` directives is summarized as follows:

`count` [*limit*]
`create` *vsnlist volgrpspec* [*settings*]

```
delete selection [limit]
dump selection [limit]
help
list selection [limit] [format]
load filename
quit (or q, or exit)
repair selection
select selection [limit]
set format
update selection [limit] to settings
verify selection
```

The *volgrpspec* parameter consists of the keyword `volgrp` (or `vg`), followed by a value for that keyword.

The value for *vsnlist* may be a single 6-character volume serial number (VSN) or a range of VSNs separated by the hyphen (`-`) character. A VSN string is case insensitive and may consist entirely of letters, entirely of digits, or a series of letters followed by digits. In a range of VSNs, the first must be lexically less than the second.

The value for *selection* may be one of the following:

- A *vsnlist* or range of VSNs in the form *vsn*`[-`*vsn*`]`. *vsn*`-` specifies all records starting with *vsn*, and `-`*vsn* specifies all records up to *vsn*.
- A period (`.`), which recalls the previous selection.
- The name of one of the flags in the keyword list that follows in this section.
- One of the words `all`, `used`, `empty`, or `partial` or any of the *hold flags (hflags)*, whose meanings are as follows:

Flag	Description
`all`	Specifies all volumes in the LS database
`empty`	Specifies all volumes in which data left is `0`
`partial`	Specifies used volumes in which `hfull` is `off`
`used`	Specifies all volumes in which data written is not `0`

- An expression involving *vsnlists*, field-value comparisons, `and`, `or`, or parentheses.

  A field value comparison may use the following to compare a field keyword to an appropriate value:

< (less than)
> (greater than)
= (equal)
!= (not equal)
<= (less than or equal to)
>= (greater than or equal to)

The syntax for *selection* is as follows:

```
selection       ::=   or-expr
or-expr         ::=   and-expr [ or or-expr ]
and-expr        ::=   nested-expr [ and or-expr ]
nested-expr     ::=   comparison | ( or-expr )
comparison      ::=   vsnlist | field-keyword op field-value
op              ::=   < | > | = | != | >= | <=
vsn-range       ::=   vsn [ - vsn] | [vsn - [vsn]] | key-macro
key-macro       ::=   all | empty | used | partial | flag(s)
field-keyword   ::=   name or abbreviation of the record field
field-value     ::=   appropriate value for the field
vsnlist         ::=   character representation of the volume serial number
```

Thus valid *selections* could be any of the following:

```
tape01-tape02
tape50-
-vsn900
all
hoa or hro
used and hfull=off
datawritten>0 and hfull=off
. and eotchunk>3000 and (eotchunk<3500 or hfree=on)
hfull and threshold<30
```

`dmvoladm` Field Keywords

You can use the *field* keywords listed below as part of the following:

- A *selection* parameter to select records
- A *format* parameter
- A *settings* parameter to specify new values for a field, in which case you must specify a keyword-value pair

A keyword-value pair consists of a keyword followed by white space and then a value. When specifying new values for fields, some of the keywords are valid only if you also specify the `-u` (unsafe) option:

Keyword	Description
`blocksize` (`bs`)	Specifies the data block size in bytes when the tape was first written; an integer. This keyword is used only when mounting volumes with existing valid data. When an empty volume is first written, the VG uses the default value for the volume type, unless it is overridden by a value in the `BLOCK_SIZE` parameter for the drive group in the DMF configuration file. This is valid only in unsafe (`-u`) mode.
`chunksleft` (`cl`)	Specifies the number of active chunks on the volume; an integer. This is valid only in unsafe (`-u`) mode.
`dataleft` (`dl`)	Specifies the number of bytes of active data on the volume. You specify this number as an integer, but for readability purposes it is displayed in megabytes (MB). This is valid only in unsafe (`-u`) mode.
`datawritten` (`dw`)	Specifies the maximum number of bytes ever written to the volume. You specify this number as an integer, but for readability purposes it is displayed in MB. This is valid only in unsafe (`-u`) mode.
`eotblockid` (`eb`)	Specifies the block ID of the chunk containing the end-of-volume marker (historically known as *EOT* for *end-of-tape*); a hexadecimal integer. This is valid only in unsafe (`-u`) mode.
`eotchunk` (`ec`)	Specifies the number of the chunk containing the end-of-volume marker; an integer. This is valid only in unsafe (`-u`) mode.
`eotpos` (`ep`)	Specifies the absolute position of the end-of-volume marker zone in the form *integer*`/`*hexadecimal-integer*, designating a zone number and block ID. A value of zero is used for *hexadecimal-integer* if no block ID is known. *integer* the same as `eotzone`, and *hexadecimal-integer* is the same as `eotblockid`. This is valid only in unsafe (`-u`) mode.

`eotzone` (`ez`)	Specifies the number of the zone containing the end-of-volume marker; an integer. This is valid only in unsafe (`-u`) mode.
`hflags` (`hf`)	Specifies the flags associated with the record. See the description of *flags* keywords. Not valid as part of a *settings* parameter.
`label` (`lb`)	Specifies the label type: • `al` (ANSI label) • `nl` (no label, not allowed for COPAN MAID or JBFS) • `sl` (standard label for IBM tapes) The default is `al`.
`tapesize` (`ts`)	Specifies the estimated capacity in bytes; an integer. The default is `0`. This field must be accurately set in order to estimate remaining capacity.
`threshold` (`th`)	Specifies the ratio of `dataleft` to `datawritten` as a percentage. This field is valid only as part of a *selection* parameter.
`upage` (`ua`)	Specifies the date and time of the last update to the volume's database record. The same as for `update`, except that it is expressed as *age*. This is not valid as part of a *settings* parameter.
`update` (`ud`)	Specifies the date and time of the last update to the volume's database record, expressed as an integer that reflects raw UNIX or Linux time. This is not valid as part of a *settings* parameter.
`version` (`v`)	Specifies the DMF media format version, an integer. This is valid only in unsafe (`-u`) mode.
`volgrp` (`vg`)	Specifies the VG or allocation group. Changing this field in an existing record is valid only in unsafe (`-u`) mode.
`wfage` (`wa`)	Specifies the date and time that the volume was written to or freed for reuse. The same as for `wfdate`, except that it is expressed as *age*. This is valid only in unsafe (`-u`) mode.

`wfdate` (`wd`)	Specifies the date and time that the volume was written to or freed for reuse, expressed as an integer that reflects raw UNIX or Linux time. This is valid only in unsafe (`-u`) mode.

The date field keywords (`update` and `wfdate`) have a value of one of the following:

- `now`
- UNIX or Linux raw time (seconds since January 1, 1970)

These keywords display their value as raw time. The value comparison `>` used with the date keywords means newer than the value given. For example, `>36000` is newer than 10AM on January 1, 1970, and `>852081200` is newer than 10AM on January 1, 1997.

The age field keywords (`upage` and `wfage`) let you express time as *age* as a string.

The age keywords display their value as an integer followed by the following:

`w` (weeks)
`d` (days)
`h` (hours)
`m` (minutes)
`s` (seconds)

For example, `8w12d7h16m20s` means 8 weeks, 12 days, 7 hours, 16 minutes, and 20 seconds old.

The comparison `>` used with the age keywords means older than the value given (that is, `>5d` is older than 5 days).

The *limit* parameter in a directive limits the records acted upon. It consists of one of the following keywords followed by white space and then a value. The abbreviation for the keyword is given in parentheses following its name, if one exists:

Keyword	**Description**
`datalimit` (no abbreviation)	Specifies a value in bytes. The directive stops when the sum of `dataleft` of the volumes processed so far exceeds this value.
`recordlimit` (`rl`)	Specifies a number of records; an integer. The directive stops when the number of volumes processed equals this value.

`recordorder` (`ro`)	Specifies the order that records are scanned; may be either `data` or `vsn`. `vsn` specifies that the records are scanned in ascending order of the chunk VSN. `data` specifies that the records are scanned in the order in which they are found in the LS database, which is fastest but essentially unordered.

The *format* parameter in a directive consists of the word `format` followed by white space and then one of the following:

- The word `default`
- The word `keyword` (which suppresses the headings and is intended for parsing by a program or script)
- A list of field and or flag keywords that may be delimited by colons or spaces (spaces require the use of quoting)

The VSN is always included as the first field and need not be specified.

The *flag* keywords listed below can be used to change the settings of the *hold flags (hflags)*. They can also be used as part of selection or format parameters:

Keyword	**Description**
`herr` (`he`)	Indicates an LS database inconsistency for this volume. It is displayed as `e--------`.
`hextern` (`hx`)	*(OpenVault only)* Indicates that the volume was not in the library. It is displayed as `--------x`. If the `hextern` flag is set, the volume cannot be merged. DMF periodically obtains from OpenVault the list of volumes that are currently in the library. If a volume is not in the library, DMF will set its `hextern` flag; if the volume is added to the library, DMF will eventually clear its `hextern` flag, but up to 75 minutes can pass before DMF notices this change. This flag does not apply to COPAN MAID configurations.
`hflags` (no abbreviation)	(Not valid as part of a *settings* parameter.) Shows the complete set of hold flags as a 9–character string. Each flag has a specific position and alphabetic value. If the flag is off, a hyphen(`-`) is displayed in its position; if the flag is on, the alphabetic character is displayed in that position.

`hfree` (no abbreviation)	Indicates that the volume has no active data and is available for reuse after `HFREE_TIME` has expired, displayed as `-f-------`. See the `dmf.conf`(5) man page for information about the `HFREE_TIME` configuration parameter. This is valid only in unsafe (`-u`) mode.
`hfull` (`hu`)	Indicates that the volume cannot hold any more data; displayed as `------u--`.
`hlock` (`hl`)	Indicates that the volume cannot be used for either input or output. This is a transient condition; the flag will be cleared by the LS after `REINSTATE_VOLUME_DELAY` has expired and at LS startup. Displayed as `----l----`.
`hoa` (`ho`)	Indicates that the volume is not to be used for either input or output, displayed as `--o------`. This value is only set or cleared by the site administrator.
`hro` (`hr`)	Indicates that the volume is read-only, displayed as `---r-----`; this inhibits the LS from using the volume for output. This value is only set or cleared by the site administrator.
`hsite1` (`h1`)	Reserved for site use; ignored by DMF. Not normally displayed; see the `dmvoladm`(8) man page for details. `hsite2`, `hsite3`, and `hsite4` are also available.
`hsparse` (`hs`)	Indicates that the volume is considered sparse and thus a candidate for a volume merge operation, displayed as `-------s-`.
`hvfy` (`hv`)	Indicates that this tape should be tested and/or replaced when next empty; until that time, it is read-only. Displayed as `----v----`. This value is set by DMF but only cleared by the site administrator.

For any field that takes a byte count, you may append one of the following letters (in either uppercase or lowercase) to the integer to indicate that the value is to be multiplied (all of which are powers of 1000, not 1024):

`k` or `K` for 1 thousand
`m` or `M` for 1 million
`g` or `G` for 1 billion

For information about the role of the dmvoladm command in LS database recovery, see "Recovery and Restoration of DMF Database" on page 646. For details about dmvoladm syntax, see the man page.

dmvoladm Text Field Order

The text field order for volume records generated by the dmdump(8), dmdumpj(8), and the dump directive in dmvoladm is listed below. This is the format expected by the load directives in dmvoladm:

1. V (indicates the volume record type)
2. vsn
3. volgrp
4. lbtype
5. capacity
6. blocksize
7. hflags (in octal)
8. version
9. datawritten
10. eotchunk
11. eotposition (eotzone/eotblockid) (in hexadecimal)
12. dataleft
13. chunksleft
14. wfdate
15. update
16. id (in octal). This field indicates the type of process that last updated the record.

dmvoladm Examples

Example 14-3 dmvoladm update Directive

The following unsets the hlock for C02M02, indicating that the volume can now be used for either input or output:

```
adm 8>update C02M02 to hlock off
```

Example 14-4 dmvoladm list Directive to Show Information for Multiple VSNs

The following example illustrates the default format for the list directive when using an LS. The column marked HFLAGS uses a format similar to the ls -l command in that each letter has an assigned position and its presence indicates that the flag is on. The positions spell out the string eforvlusx, representing herr, hfree, hoa, hro, hvfy, hlock, hfull, hsparse, and hextern.

```
adm 1> list 000683-000703
                                          DATA    EOT   EOT             WR/FR
VSN      VOLGRP LB      DATA LEFT        WRITTEN  CHUNK  ZONE   HFLAGS      AGE
-------------------------------------------------------------------------------
000683   vg8a01 al       0.000000       0.000000      1     1  ---------     3d
000700   vg8a00 al     267.539255     287.610294    124     7  ------u--     2d
000701   vg8a00 al     288.342795     308.147798    136     8  ------u--     2d
000702   vg8a00 al     255.718902     288.302830    120     7  ------u--     2d
000703      ag8 al       0.000000       0.000000      1     1  ---------     3d
```

Example 14-5 dmvoladm list Directive to Show Volumes with a Specific Flag

The following example illustrates using the list command to show only volumes having their hfull flag set:

```
adm 1>list hfull
                                          DATA    EOT   EOT             WR/FR
VSN      VOLGRP LB      DATA LEFT        WRITTEN  CHUNK  ZONE   HFLAGS      AGE
-------------------------------------------------------------------------------
000701   vg8a00 al     288.342795     308.147798    136     8  ------u--     2d
000702   vg8a00 al     255.718902     288.302830    120     7  ------u--     2d
000704   vg8a00 al     252.294122     292.271410    119     7  ------u--     2d
000705   vg8a00 al     250.207666     304.603059    143     7  ------u--     2d
000706   vg8a00 al     265.213875     289.200534    144     7  ------u--     2d
000707   vg8a00 al     278.744448     310.408119    140     7  ------u--     2d
000708   vg8a00 al     260.827748     295.956588    136     7  ------u--     2d
```

```
000709   vg8a00 al     253.481897     283.615678    138     8  ------u--     2d
000710   vg8a00 al     265.100985     291.243235    141     7  ------u--     2d
000711   vg8a00 al     276.288446     305.782035    144     7  ------u--     2d
000712   vg8a00 al     250.415786     275.606243    138     7  ------u--     2d
000716   vg8a00 al     287.964765     304.321543    144     7  ------u--     2d
000717   vg8a00 al     280.795058     287.084534    144     7  ------u--     2d
000718   vg8a00 al       0.000415     300.852018    180    27  ------u--     3d
003127   vg9a01 al     417.383784     461.535047    209    10  ------u--     2d
003128   vg9a01 al     427.773679     460.716741    229    11  ------u--     2d
```

Example 14-6 `dmvoladm list` Directive to Customize a List of Fields

The following example shows one way you can customize the list format to show only the fields that you want to see.

```
adm 21>list S03232-S03254 format "eotchunk eotzone eotpos"
          EOT   EOT
VSN     CHUNK  ZONE    EOTPOS
--------------------------------
S03232     10     2 2/4294967295
S03233      2     2 2/4294967295
S03234    598     2 2/4294967295
S03235     18     2 2/4294967295
S03236     38     2 2/4294967295
S03237     92     2 2/4294967295
S03238      1     1 1/4294967295
S03239      1     1 1/4294967295
S03240      1     1 1/4294967295
S03241    325     2 2/4294967295
S03242     81     2 2/4294967295
S03243     26     2 2/4294967295
S03244      1     1 1/4294967295
S03245     26     2 2/4294967295
S03246      5     2 2/4294967295
S03247    186     2 2/4294967295
S03248     17     2 2/4294967295
S03249    526     2 2/4294967295
S03250      1     1 1/4294967295
S03251    533     2 2/4294967295
S03252    157    17 17/2147483648
S03253    636     2 2/4294967295
```

```
S03254     38     2 2/4294967295
```

Another way to accomplish this is to use the `set format` command with the same keyword list.

Example 14-7 `dmvoladm list` Directive to Show Multiple Flags

The following example gives a convenient way to show the setting of multiple flags:

```
adm 23>list 003232-003254 format "hfree hfull hlock hoa hro"
       hfree hfull hlock hoa hro
VSN
--------------------------------
003232  off    on    off off off
003233  off   off    off off off
003234  off   off    off off off
003235  off   off    off off off
003236  off    on    off off off
003237  off    on    off off off
003238  off    on    off off off
003239  off    on    off off off
003240  off   off    off off off
003241  off    on    off off off
003242  off    on    off off off
003243  off   off    off off off
003244  off   off    off off off
003245  off    on    off off off
003246  off   off    off off off
003247  off    on    off off off
003248  off    on    off off  on
003249   on   off    off off  on
003250   on   off    off off  on
003251   on   off    off off  on
003252   on   off    off off  on
003253  off    on    off off  on
003254  off    on    off off  on
```

Example 14-8 `dmvoladm list` Directive to Display Volumes Assigned to a VG

The following example shows how to display only those volumes assigned to the VG named `vg9a00`.

```
adm 3>list vg=vg9a00
                                          DATA    EOT   EOT             WR/FR
VSN      VOLGRP LB      DATA LEFT        WRITTEN  CHUNK  ZONE   HFLAGS      AGE
-------------------------------------------------------------------------------
003210   vg9a00 al       1.048576       1.048576      3     2  ---------    11d
003282   vg9a00 al      11.534336      11.534336     13     2  ---------     7d
```

`dmatread` Command

You can use `dmatread`(8) to copy all or part of the data from a migrated file back to disk. You might want to do this if, for example, a user accidentally deleted a file and did not discover that the deletion had occurred until after the database entries had been removed by the hard delete procedure. Using backup copies of the databases from before the hard delete was performed, `dmatread` can restore the data to disk, assuming that the tape volume has not been reused in the meantime.

Example 14-9 Restoring Hard-deleted Files Using `dmatread`

To copy migrated files back to disk, perform the following steps:

1. Determine the BFID of the file you want to restore. You can use backup copies of `dmdlog` or your `dbrec.dat` files, or a restored backup copy of the deleted file's inode (and the `dmattr` command).
2. Using backup copies of the LS database, execute a `dmatread`(8) command similar to the following:

   ```
   dmatread -p /a/dmbackup -B 342984C50000000000084155
   ```

   `342984C50000000000084155` is the BFID of the file to be restored, and `/a/dmbackup` is the directory containing the backup copies of the LS database. Your file will be restored to the current directory as `B342984C50000000000084155`.

Note: DMF does not know the original name of the file; you must manually move the restored data to the appropriate file.

If you have access to chunk and VSN information for the file to be restored, you can use the `dmatread -c` and `-v` options and avoid using backup copies of the LS database. In this case, `dmatread` will issue messages indicating that the chunk is not found in the current LS database, but it will continue with the request and restore the file as described in this example.

`dmatsnf` Command

You can use `dmatsnf`(8) to verify the readability of the LS volumes or to audit their contents. The `dmatsnf` script is a wrapper around the `dmatsnfb` binary. Both the script and the binary are installed on the DMF server, but only the binary is installed on parallel data-mover nodes. In most cases, you will execute `dmatsnf`. When using `dmatsnf`, in most cases you will only need to specify the VSNs, the volume's volume group, and the type of reports desired. For more information about how the binary and script work together, see the `dmatsnf`(8) man page.

Note: The `dmatsnf`(8) and `dmatread`(8) commands verify the integrity of the library server (LS) volumes on MAID shelves and recover data from them. For those volumes that are mountable only on a parallel data-mover node, use of these commands is simplified if there is passwordless `ssh`(8) connection from the DMF server to the parallel data-mover node. For more information about these commands, see their man pages.

You can also use `dmatsnf` to verify one or more volumes against the LS database or to generate journal entries, which you can add to the LS database by using the `load` directive in `dmvoladm` and `dmcatadm`.

You may also generate text database records that you can apply to the LS database using the `load` directive in `dmcatadm` and `dmvoladm`. You can use the text records to add the contents of a few volumes to the LS database (however, this is impractical for large numbers of volumes).

`dmaudit verifymsp` Command

You can use the `verifymsp` option of the `dmaudit`(8) command to check the consistency of the daemon database and LS database after an MSP, LS, DMF daemon, or system failure. This command captures the database files and compares the contents of the daemon database with each LS database. Any problems are reported to standard output, but no attempt is made to repair them.

You can also perform a similar function directly using `dmatvfy`(8) after taking a snapshot.

Cloud MSP

This section discusses the following:

- "Cloud MSP Overview" on page 569
- "Cloud MSP Processing of Requests" on page 569
- "Cloud MSP Activity Log" on page 571
- "Cloud MSP Messages" on page 571
- "Cloud MSP Limitations and Considerations" on page 572

Cloud MSP Overview

The cloud MSP allows the DMF daemon to manage data by moving it to a cloud storage service:

- Scality RING private cloud
- Amazon Simple Storage Service (S3) public cloud
- Other products that present a service interface that is compatible with S3, in a private cloud

The cloud MSP does not need a private database to operate; all information necessary to retrieve offline files is kept in the daemon database, DMF configuration file, and login information file. The login information file contains a password that must be kept private. As a safeguard, the MSP will not operate if the login information file is readable by anyone other than the system administrator.

Cloud MSP Processing of Requests

The cloud MSP is always waiting for requests to arrive from the DMF daemon. To improve efficiency, it holds `PUT` and `DELETE` requests briefly and groups similar requests together into a single cloud session:

- No `PUT` request will be held longer than 60 seconds

- No DELETE request will be held longer than 5 seconds

Note: GET requests are not held.

The cloud MSP will stop holding requests if it has a large amount of work to do (more than 1024 individual files or 8 MB of data). The cloud MSP also limits the number of cloud services that can be active at once and the rate at which new sessions can be initiated.

After a request has been held for the appropriate amount of time, it enters a ready state. Processing usually begins immediately, but may be delayed if resources are not available.

The following limits affect the maximum number of requests that can be processed:

- An administrator-controlled limit on the maximum number of concurrent cloud-service sessions per MSP (CHILD_MAXIMUM).
- An administrator-controlled limit on the number of child processes that are guaranteed to be available for processing delete requests (GUARANTEED_DELETES).
- An administrator-controlled limit on the number of child processes that are guaranteed to be available for processing dmget(1) requests (GUARANTEED_GETS).
- A system-imposed limit to the number of cloud-service sessions in any 60-second period. This limit is seldom a concern because of the MSP's ability to transfer many files in one session.

The following steps occur:

1. Requests are processed by forking off a child process.
2. The parent process immediately resumes waiting for requests to arrive from the DMF daemon.
3. The child process attempts to initiate a cloud-service session on the backend. If multiple IP addresses are specified with the CLOUD_HOST configuration parameter, one will be chosen at random until a connection is established (see "msp Object Parameters for a Cloud MSP" on page 449).
4. After a connection is established, the child process provides any required user name and password information to the cloud server.

5. Requests are distributed to the running child processes on a first-come, first-served basis. Requests are not batched; children handle at most one request at a time. Requests handed to any particular child process might be freely intermixed; a given child may handle any combination of `PUT`, `GET`, or `DELETE` operations (even if `GUARANTEED_GETS` or `GUARANTEED_DELETES` is set in the DMF configuration file).

   The stored files are not verbatim copies of the user files. They are stored using the same format used when writing volumes.

Cloud MSP Activity Log

All DMF MSPs maintain log files named `msplog.`*yyyymmdd* in the MSP spool directory which, by default, is *SPOOL_DIR*`/`*mspname*. *SPOOL_DIR* is configured in the `base` object of the configuration file; *mspname* is the name of the MSP in the `daemon` object of the configuration file; *yyyymmdd* is the current year, month, and day.

The activity log shows the arrival of new requests, the successful completion of requests, failed requests, creation and deletion of child processes, and all cloud transactions. Sensitive information (passwords and account information) does not appear in the activity log. In addition, the MSP lists the contents of its internal queues in its activity log if it is given an `INTERRUPT` signal.

Note: Because the cloud MSP will continue to create log files without limit, you must remove obsolete files periodically by configuring the `run_remove_logs` task in the configuration file, as described in "`taskgroup` Object" on page 304.

Cloud MSP Messages

The MSP also recognizes and handles the following messages issued from the DMF daemon:

Message	Description
`CANCEL`	Issued when a previously requested action is no longer necessary, for example, when a file being migrated with a `PUT` request is removed. The MSP is able to cancel a request if it is being held or if it is waiting for resources. A request that has begun processing cannot be canceled and will run to normal completion.

`FINISH` Issued during normal shutdown. When the MSP receives a `FINISH` message, it finishes all requested operations as quickly as it can and then exits.

`FLUSHALL` Issued in response to the `dmdidle`(8) command. When the MSP receives a `FLUSHALL` message, it finishes all requested operations as quickly as it can.

Cloud MSP Limitations and Considerations

Note the following:

- Security issues:
  - SSL and HTTPS are not supported.
  - Migrated data is not encrypted.
  - Scality supports only a clear-text password.
  - HTTP Basic Authentication is not enabled by default on Scality systems.
- Support issues:
  - The cloud MSP, Scality, and S3 support only IPv4.
  - The cloud MSP does not support Amazon Glacier.
- A given cloud can be a single point of failure, therefore redundant copies within one cloud do not sufficiently protect against data loss. SGI therefore highly recommends that you migrate data to a second location (to another cloud instance, tape, or disk).
- The cloud MSP breaks up files into 1–MiB chunks. To potentially minimize access times, you may want to use substitution strings to randomize the order of keys (rather than using sequentially named keys); see "`NAME_FORMAT` Details" on page 446.
- The `dmatread`(8) and `dmatsnf`(8) utilities do not work on cloud storage.
- The `dmcheck`(8) command and the validation operation in DMF Manager verify that at least one cloud host is available.
- If the cloud must be restored to a previous state, inconsistencies may arise: remote files that reappear after being deleted are never removed, and remote files that

disappear unexpectedly result in data loss. There is presently no way to detect these inconsistencies. You should avoid situations that require the cloud to be restored to a previous state.

FTP MSP

This section discusses the following:

- "FTP MSP Overview" on page 573
- "FTP MSP Processing of Requests" on page 573
- "FTP MSP Activity Log" on page 575
- "FTP MSP Messages" on page 576
- "FTP MSP Limitations and Considerations" on page 576

FTP MSP Overview

The FTP MSP allows the DMF daemon to manage data by moving it to a remote machine. Data is moved to and from the remote machine with the protocol described in RFC 959 (FTP). The remote machine must understand this specific protocol.

The FTP MSP does not need a private database to operate; all information necessary to retrieve offline files is kept in the daemon database, DMF configuration file, and login information file. The login information file contains a password that must be kept private. As a safeguard, the MSP will not operate if the login information file is readable by anyone other than the system administrator.

FTP MSP Processing of Requests

The FTP MSP is always waiting for requests to arrive from the DMF daemon. To improve efficiency, it holds PUT and DELETE requests briefly and groups similar requests together into a single FTP session:

- No PUT request will be held longer than 60 seconds
- No DELETE request will be held longer than 5 seconds

Note: GET requests are not held.

The FTP MSP will stop holding requests if it has a large amount of work to do (more than 1024 individual files or 8 MB of data). The FTP MSP also limits the number of FTP sessions that can be active at once and the rate at which new sessions can be initiated.

After a request has been held for the appropriate amount of time, it enters a ready state. Processing usually begins immediately, but may be delayed if resources are not available.

The following limits affect the maximum number of requests that can be processed:

- An administrator-controlled limit on the maximum number of concurrent FTP sessions per MSP (CHILD_MAXIMUM).
- An administrator-controlled limit on the number of child processes that are guaranteed to be available for processing delete requests (GUARANTEED_DELETES).
- An administrator-controlled limit on the number of child processes that are guaranteed to be available for processing dmget(1) requests (GUARANTEED_GETS).
- A system-imposed limit of 85 FTP sessions in any 60-second period. This limit is seldom a concern because of the MSP's ability to transfer many files in one session.

The following steps occur:

1. Requests are processed by forking off a child process.
2. The parent process immediately resumes waiting for requests to arrive from the DMF daemon.
3. The child process attempts to initiate an FTP session on the remote FTP server.
4. If the remote machine has multiple IP addresses in the name system, only the first is tried.
5. After a connection is established, the child process provides any required user name, password, account, and default directory information to the remote FTP server.
6. PUT, GET, or DELETE operations are then performed as requested by the DMF daemon.

Binary transfer mode is used for all data transfer. Passive mode (the PASV or EPSV commands) will be tried, but the child process will fall back to the active mode (the PORT or EPRT commands). Block mode (the MODE B command) will be tried in order to allow the data connection to be used for multiple transfers, but the child process will fall back to a single data connection per transfer if the server does not support MODE B, which is the most likely case.

7. Requests are distributed to the running child processes on a first-come, first-served basis. Requests are not batched; children handle at most one request at a time. Requests handed to any particular child process might be freely intermixed; a given child may handle any combination of PUT, GET, or DELETE operations (even if GUARANTEED_GETS or GUARANTEED_DELETES is set in the DMF configuration file).

   The stored files are not verbatim copies of the user files. They are stored using the same format used to write volumes, and you can use MSP utilities such as dmatread and dmatsnf to access the data in them.

FTP MSP Activity Log

All DMF MSPs maintain log files named msplog.*yyyymmdd* in the MSP spool directory which, by default, is *SPOOL_DIR*/*mspname*. *SPOOL_DIR* is configured in the base object of the configuration file; *mspname* is the name of the MSP in the daemon object of the configuration file; *yyyymmdd* is the current year, month, and day.

The activity log shows the arrival of new requests, the successful completion of requests, failed requests, creation and deletion of child processes, and all FTP transactions. Sensitive information (passwords and account information) does not appear in the activity log. In addition, the MSP lists the contents of its internal queues in its activity log if it is given a SIGINT signal.

Note: Because the FTP MSP will continue to create log files without limit, you must remove obsolete files periodically by configuring the run_remove_logs task in the configuration file, as described in "taskgroup Object" on page 304.

FTP MSP Messages

The MSP also recognizes and handles the following messages issued from the DMF daemon:

Message	Description
CANCEL	Issued when a previously requested action is no longer necessary, for example, when a file being migrated with a PUT request is removed. The MSP is able to cancel a request if it is being held or if it is waiting for resources. A request that has begun processing cannot be canceled and will run to normal completion.
FINISH	Issued during normal shutdown. When the MSP receives a FINISH message, it finishes all requested operations as quickly as it can and then exits.
FLUSHALL	Issued in response to the dmdidle(8) command. When the MSP receives a FLUSHALL message, it finishes all requested operations as quickly as it can.

FTP MSP Limitations and Considerations

Note the following:

- The FTP MSP supports only a clear-text password.
- If the remote filesystem must be restored to a previous state, inconsistencies may arise: remote files that reappear after being deleted are never removed, and remote files that disappear unexpectedly result in data loss. There is presently no way to detect these inconsistencies. You should avoid situations that require the remote filesystem to be restored to a previous state.

Disk MSP

This section discusses the following:

- "Disk MSP Overview" on page 577
- "Disk MSP Processing of Requests" on page 577
- "Disk MSP Activity Log" on page 578

Disk MSP Overview

The disk MSP (`dmdskmsp`) migrates data into a directory that is accessed on the current system. It uses POSIX file interfaces to open, read, write, and close files. The directory may be NFS-mounted, unless the disk MSP is configured as a disk cache manager MSP (see "DCM MSP" on page 579). The data is read and written with `root` (UID 0) privileges. By default, `dmdskmsp` stores the data in a blocked format that is specific to DMF, which allows the MSP to do the following:

- Keep metadata with a file
- Keep sparse files sparse when they are recalled
- Verify that a file is intact on recall

The disk MSP does not need a private database to operate; all information necessary to retrieve offline files is kept in the daemon database and DMF configuration file.

The disk MSP may also be used as an import MSP. In this case, it only permits recalls and copies the data unchanged for a recall.

Disk MSP Processing of Requests

The disk MSP is always waiting for requests to arrive from the DMF daemon. To improve efficiency, it holds `PUT` and `DELETE` requests briefly and groups similar requests together into a single session:

- No `PUT` request will be held longer than 60 seconds
- No `DELETE` request will be held longer than 5 seconds

Note: `GET` requests are not held.

The disk MSP will stop holding requests if it has a large amount of work to do (more than 1024 individual files or 8 MB of data).

After a request has been held for the appropriate amount of time, it enters a ready state. Processing usually begins immediately, but may be delayed if resources are not available.

The following administrator-controlled limits affect the maximum number of requests that can be processed:

- Maximum number of concurrent operations per MSP (CHILD_MAXIMUM)
- Number of child processes that are guaranteed to be available for processing delete requests (GUARANTEED_DELETES)
- Number of child processes that are guaranteed to be available for processing dmget(1) requests (GUARANTEED_GETS)

Requests are processed by forking off a child process. The parent process immediately resumes waiting for requests to arrive from the DMF daemon.

Requests are shared out amongst the running child processes on a first-come, first-served basis. Requests are not batched; children handle at most one request at a time. Requests handed to any particular child process might be freely intermixed; a given child may handle any combination of PUT, GET, or DELETE operations (even if GUARANTEED_GETS or GUARANTEED_DELETES is set in the DMF configuration file).

The stored files are not verbatim copies of the user files. They are stored using the same format used to write tapes, and you can use MSP utilities such as dmatread and dmatsnf to access the data in them.

Disk MSP Activity Log

All DMF MSPs maintain log files named msplog.*yyyymmdd* in the MSP spool directory which, by default, is *SPOOL_DIR*/*mspname*. *SPOOL_DIR* is configured in the base object of the configuration file; *mspname* is the name of the MSP in the daemon object of the configuration file; *yyyymmdd* is the current year, month, and day).

The log file shows the arrival of new requests, the successful completion of requests, failed requests, and creation and deletion of child processes. In addition, the MSP lists the contents of its internal queues in its activity log if it is given an INTERRUPT signal.

Note: Because the disk MSP will continue to create log files without limit, you must remove obsolete files periodically by configuring the run_remove_logs task in the configuration file, as described in "taskgroup Object" on page 304.

DCM MSP

The *Disk cache manager (DCM) MSP* is the disk MSP configured for *n*–tier capability using a dedicated filesystem as a cache. The DCM MSP provides fast access for files whose activity levels remain high while also providing migration to tape/MAID/JBFS for those files requiring less frequent access.

To allow the disk store that is managed by the disk MSP to function as a dynamically managed cache (as opposed to a static store), the DCM MSP creates and maintains a filesystem attribute on each file that is created in the MSP `STORE_DIRECTORY`. This attribute is used by the `dmdskfree` process to evaluate files for downward migration and for possible removal from the disk cache. For this reason, the DCM MSP `STORE_DIRECTORY` must be a local XFS or CXFS filesystem mount point with DMAPI enabled.

The DCM MSP supports *dual-resident state*, in which files reside in the cache and also in a lower VG. This provides the access speed of a disk file, but allows that cache file to be quickly released without the need to first write it to tape/MAID/JBFS. This is directly analogous to the concept of a dual-state file in the standard managed filesystem.

Automated movement in the opposite direction (from tape/MAID/JBFS back to the cache) is not available. Any recalls of files that no longer have copies held in the cache will come directly from tape/MAID/JBFS; they are not recalled via the cache and they can only be restored to the cache by an explicit `dmmove`(8) command.

`dmdskvfy` Command

The `dmdskvfy` command verifies that copies of migrated files in DCM and disk MSPs are consistent with the daemon database entries that refer to them.

Moving Migrated Data

The `dmmove`(8) command moves copies of offline or dual-state files to a specified set of MSPs, VGs, or MGs. The options specified on command line indicate which targets are to contain migrated copies of a file after the move process is completed. All other migrated copies are hard-deleted unless you use the `dmmove -d` to select specific copies for deletion.

If a file's migrated state is offline, dmmove recalls the file to disk and then remigrates it to the specified targets. (The one exception to this is that if a disk cache manager disk MSP copy exists, the file will be moved directly from that file copy.) The file is recalled to a scratch filesystem that is specified by the MOVE_FS configuration parameter. When the migration process is complete, the online copy is removed. If the file is dual-state, dmmove does not need to recall the file first, but instead uses the existing online copy.

The dmselect(8) command can be used to determine which files you want to move. dmselect selects files based on age, size, ownership, and MSP criteria. The output from the dmselect command can be as input to the dmmove command. The dmmove command also accepts a list of pathnames as input.

See the dmselect(8) and dmmove(8) man pages for all of the possible options and further information.

LS Error Analysis and Avoidance

The drive group component of the LS monitors media use, analyzes failures, and uses this information to avoid future errors.

The drive group component can react to some failures without looking for any patterns of behavior. Among these are the following:

- Mounting service failure. If the mounting service is TMF, by default, DMF makes one attempt to restart it. If this attempt does not succeed, DMF notifies the administrator by e-mail and waits for the administrator's intervention. When TMF is back again, DMF resets the auto-restart flag so that if TMF fails again, it will once again make one attempt to restart it.

  If OpenVault is the mounting service, by default, no attempt is made to restart it. Instead, an e-mail is sent to the administrator.

  A site can set the number of automatic restart attempts by using the drive group's MAX_MS_RESTARTS configuration parameter, but caution and thorough testing are advised. There are many possible failure modes for a mounting service, and automated restarts might not always be appropriate.

- Volume is not in the library. Obviously, this problem will not be fixed by trying again. To prevent further access, the volume is locked by setting the HLOCK flag, as described below, and the user requests that triggered the access attempt are

retried on another tape, if possible; otherwise, they are aborted. The administrator is notified by e-mail.

- For TMF only, a tape mount was cancelled by an operator or administrator. Although the user requests are retried or aborted, the volume is not disabled. If the volume were disabled, it would be inaccessible for a period of time (default 24 hours) unless `dmvoladm` were used to preempt this delay. All operators do not necessarily have access to the `dmvoladm` command.

  Because the reason for the cancellation is unknown to DMF, repeated requests for the same volume are quite possible, and the operator might have to cancel each one.

The drive group handles other types of failure by examining the recent history of the volume and the drive that was used. The drive group maintains records of past I/O errors and uses these to control the way it reacts to future errors.

For example, if a tape has been unusable several times in a row, even though different drives were used, the drive group concludes that the problem most likely involves the tape rather than the drive. Therefore, it suspends use of that tape, forcing DMF to migrate to a different tape in that VG or to recall the file from another tape held by a different VG. This suspension is usually done by setting the `HLOCK` flag in the tape's entry in the VOL record of the LS database. This makes the tape inaccessible to the VG for both reading and writing until it is automatically cleared after `REINSTATE_VOLUME_DELAY` minutes.

If a variety of volumes fail on a specific drive but are usable on other drives, a drive problem is likely, and the drive can be automatically configured down if permitted by the administrator's setting of `DRIVES_TO_DOWN` to a value higher than its default of zero. When a drive is configured down in this way, it is configured up again after `REINSTATE_DRIVE_DELAY` minutes.

The analyses of drive and volume errors are performed independently of each other; it is possible for one additional error to result in both the drive and the volume being disabled.

There are several reasons for reinstating drives and volumes after a delay. The most important is that the analyses of previous failures might lead to a faulty conclusion in some situations, such as when DMF is under a very light load, or when multiple failures occur concurrently. A wrong diagnosis might impact DMF's performance, and should not be accepted indefinitely. Disabling a suspected drive or volume for a while is usually enough to break any repetitive cycles of failure. If such patterns reestablish

themselves when the reinstatement occurs, the drive group will again analyze the behavior, possibly reaching a different conclusion, and again try to prevent it.

There are some variations from these general reactions. For example, if a tape with existing data on it is diagnosed as faulty when appending new data, instead of setting the HLOCK flag, the drive group sets HVFY, which results in the tape being used in a read-only mode until eventually emptied by merges or hard deletion of its files. At that time, the administrator may choose to test it and possibly replace or delete it. If it is to be returned to service, the HVFY flag should be cleared by using `dmvoladm`. Full details of these procedures are included in the email sent to the administrator at the time of the error.

If it is considered desirable to return a volume or drive to service earlier than defined in the DMF configuration, the appropriate command (`dmvoladm`, `tmconfig`, or `ov_drive`) can be safely used.

LS Drive Scheduling

When multiple VGs are requesting the use of more drives than exist in the drive group, the resource scheduler is used to decide which VGs should wait and which should be assigned the use of the drives.

The resource scheduler is only aware of volume-group activity on the drives in its drive group. This excludes activity such as XFS backups and direct media use by the system's users; this use does not prevent the LS from working properly, although it might be less than optimal.

LS Status Monitoring

You can observe the performance of the LS in two ways:

- Monitor its log file with a tool like `tail -f`, which allows an experienced administrator to follow the flow of events as they happen
- Use the resource watcher component, when enabled by use of the WATCHER parameter in the `libraryserver` configuration stanza

The resource watcher is intended to give the administrator a view of the status of an LS and some of its components. It maintains a set of text files on disk that are rewritten as events happen. These files can be found in the following directory:

SPOOL_DIR/libraryserverObjectName/_resourcewatcherObjectName

SPOOL_DIR is defined in the DMF configuration file (for example /dmf/spool), as are the names of the libraryserver and resourcewatcher objects (for example, lsname and rwname). The easiest way to find the precise path is to look in the LS log file for messages like the following:

```
rwname.config_changed:  URL of home page is file:/dmf/spool/lsname/_rwname/lsname.html
```

This message is issued at DMF startup or whenever the configuration file is altered or its modification time changes; for example, by using the touch(1) command.

The SPOOL_DIR/lsname/_rwname directory contains files with names ending in .html, which are automatically refreshing HTML files. You can access these files by using a browser running on the same machine. The following example shows an LS page that contains links to drive group pages, and they in turn have links to VG pages, if the VGs are active at the time:

```
netscape file:/dmf/spool/lsname/_rwname/lsname.html
```

If running the browser on the DMF machine is inconvenient, you can include the directory in your HTTP server configuration to allow those same pages to be accessed via the web.

This directory also contains files whose names end in .txt, designed to be parsed with programs like awk. The data format is described by comments within those files and can be compared with the equivalent HTML files.

If the format of the text ever changes, the version number will change. If the changes are incompatible with previous usage, the number before the decimal point is altered. If they are compatible, the number after the decimal point is altered.

An example of compatibility is adding extra fields to the end of existing lines or adding new lines. Programs using these files should check the version number to ensure compatibility. Also, it might be useful to check the following:

- DMF version shown by dmversion(1)
- Linux kernel version shown by uname(1)
- Linux distribution version shown by head /etc/*release

DMF Maintenance

This chapter contains the following:

- "Retaining Old DMF Daemon Log Files" on page 585
- "Retaining Old DMF Daemon Journal Files" on page 586
- "Cleaning Up Obsolete Database Entries" on page 586
- "Using `dmfill`" on page 587
- "Viewing Drive Statistics" on page 587
- "Temporarily Disabling Components" on page 589

See also:

- "Restoring Files from Backups" on page 643
- "Recovery and Restoration of DMF Database" on page 646

Retaining Old DMF Daemon Log Files

The DMF daemon generates the *SPOOL_DIR*`/daemon/dmdlog.`*yyyymmdd* log file, which contains a record of DMF activity and can be useful for problem solving for several months after creation. All MSPs and LSs generate a *SPOOL_DIR*`/`*msp_or_ls_name*`/msplog.`*yyyymmdd* log file, which also contains useful information about its activity. The LS also generates *SPOOL_DIR*`/`*ls_name*`/moverlogs/`*hostname*`/moverlog.`*yyyymmdd* log files, which also contain useful information about its activity. These log files should be retained for a period of some months. Log files more than a year old are probably not very useful.

Do not use DMF to manage the *SPOOL_DIR* filesystem.

The `dmfsmon`(8) automated space management daemon generates a log file in *SPOOL_DIR*`/daemon/autolog.`*yyyymmdd*, which is useful for analyzing problems related to space management.

To manage the log files, configure the `run_remove_logs.sh` task, which automatically deletes old log files according to a policy you set. See "`taskgroup` Object" on page 304, for more information.

Retaining Old DMF Daemon Journal Files

The DMF daemon and the LS generate journal files that are needed to recover databases in the event of filesystem damage or loss. You also configure DMF to generate backup copies of those databases on a periodic basis. You need only retain those journal files that contain records created since the oldest database backup that you keep. Although in many cases only the most recent database backup copy is sufficient, SGI recommends that you keep several generations for additional safety.

For example, if you configure DMF to generate daily database backups and retain the three most recent backup copies, then at the end of 18 July there would be backups from the 18th, 17th, and 16th. Only the journal files for those dates need be kept for recovery purposes.

To manage the journal files and the backups, configure the `run_remove_journals.sh` and `run_copy_databases.sh` tasks. These tasks automatically delete old journal files and generate backups of the databases according to a policy you set. See "`taskgroup` Object" on page 304, for more information.

Cleaning Up Obsolete Database Entries

When a file is migrated by DMF, a database record is created for each MSP/VG copy of the file. When the file is deleted or modified, those records point to copies that no longer correspond to any file in the filesystem. At this point, DMF tags these records as *soft-deleted*.

Soft-deleted database records must remain in the database as long as the original file might reappear as an offline file as the result of a full or partial filesystem restore. This amount of time is defined by the `DUMP_RETENTION` configuration parameter (see "`taskgroup` Object Parameters" on page 312).

After soft-deleted records pass the *DUMP_RETENTION* time, they are obsolete. At that point, they become candidates for *hard-deletion*, which permanently removes the records from the database. This decreases the total number of database records (for which there is a maximum of 4 billion). See "Use a Task to Perform Hard-Deletes Periodically" on page 131.

Caution: Do not hard-delete a database record until after you are sure that the corresponding file will never be restored.

Using dmfill

The dmfill(8) command allows you to fill a restored filesystem to a specified capacity by recalling offline files. When you execute xfsdump -a, only inodes are backed up for all files that have been migrated (including dual-state files). Therefore, when the filesystem is restored, only the inodes are restored, not the data. You can use dmfill in conjunction with xfsrestore to restore a corrupted filesystem to a previously valid state. dmfill recalls migrated files in the reverse order of migration until the requested fill percentage is reached or until there are no more migrated files left to recall on this filesystem.

Viewing Drive Statistics

To view statistics about drives across the DMF environment, you can use the dmtapestat(8) command as root from the DMF server. By default, dmtapestat displays the following fields (known as the *default field selection list*), in ascending order by drive name:

Field	Description
vg	Volume group
vsn	Volume serial number (VSN)
dg	Drive group
drive	Drive name
node	Node name
pid	Process ID
bytes_moved	Total number of bytes moved by the drive
op	Current drive operation
status	Status of the filesystem

For example, the following output shows that drive `C02d00` is idle:

```
# dmtapestat
VG      VSN     DG      DRIVE   NODE      PID    BYTES_MOVED  OP  STATUS
 dg_c02  C02d00             0      0            U   Idle
 dg_c02  C02d01             0      0            U   Idle
 dg_c02  C02d02             0      0            U   Idle
 dg_c02  C02d03             0      0            U   Idle
 dg_c02  C02d04             0      0            U   Idle
 dg_c02  C02d05             0      0            U   Idle
 dg_c02  C02d06             0      0            U   Idle
vg_c03  C03W00  dg_c03  C03d00  dignity2  29799  193986560    G   Waiting at zone 145
vg_c03  C03S00  dg_c03  C03d01  dignity2  29280  351272960    G   Waiting at zone 1654
vg_c03  C03L00  dg_c03  C03d02  dignity2  944    59768832     G   Waiting at zone 1407
vg_c03  C03J00  dg_c03  C03d03  dignity2  1934   9437184      G   Waiting at zone 36
vg_c03  C03G00  dg_c03  C03d04  dignity2  694    123731968    G   Waiting at zone 1687
vg_c03  C03B00  dg_c03  C03d05  dignity2  29536  229638144    G   Waiting at zone 1230
vg_c03  C03T00  dg_c03  C03d06  dignity2  677    105906176    G   Waiting at zone 321
```

You can customize the `dmtapestat` output by providing field names known to the common arena and using the following options to manipulate the display:

- To add fields to the output, in addition to the default field selection list:

  `-a` *field1*`[,`*field2*`...]`

- To change the list of fields displayed (overriding the default field selection list):

  `-c` *field1*`[,`*field2*`...]`

  The fields are displayed in the order specified.

- To sort the output by the specified fields:

  `-s` *field1*`[,`*field2*`...]`

  By default, `-s` sorts in ascending order. To specify descending order for a specific field, prefix the field with the minus ("`-`") sign. If you want to include white space, you must enclose the list of fields with quotation marks.

For example, to display the `pid`, `drive`, and `access_time` fields with the output sorted in descending order by process ID, enter the following:

```
# dmtapestat -c pid,drive,access_time -s -pid
PID    DRIVE   ACCESS_TIME
10491  C03d02  1317251582
10479  C03d05  1317251592
9985   C03d04  1317251572
9485   C03d01  1317251582
8950   C03d03  1317251590
8410   C03d06  1317251585
8384   C03d00  1317251580
0      C02d01  1317213541
0      C02d00  1317223473
0      C02d03  1317216907
0      C02d02  1317223560
0      C02d05  1317223292
0      C02d04  1317213499
0      C02d06  1317223472
```

For more information about available drive arena fields, see the `dmtapestat`(8) man page.

Temporarily Disabling Components

If you are using OpenVault, you can choose to temporarily disable a specific path to a drive (the *drive control program* or *DCP*), individual drives, or the entire library of drives. When you temporarily disable a drive in OpenVault, a mount request for that drive will block. (If you permanently disable a drive, a mount will be rejected).

Note: When you disable an OpenVault DCP, drive, or library, new mounts are immediately disabled but there might be running processes that are already using the drives. For safety, you must wait for DMF to notice that the component has been disabled and for those processes to stop before performing maintenance.

This section discusses the following:

- "Disable an OpenVault DCP" on page 590
- "Disable an OpenVault Drive" on page 591

- "Disable an OpenVault Library" on page 592
- "Disable a TMF Drive" on page 594
- "Stop the COPAN VTL" on page 595

Disable an OpenVault DCP

To temporarily disable one path to a drive (which still permits new mounts using other paths to that drive) when the drive is not in use, do the following:

1. Disable the DCP by using the following ov_dcp(8) command, where *drive_expr* and *dcp_expr* may be regular expressions:

   # **ov_dcp -T *drive_expr dcp_expr***

   For example, to disable the DCP t200d2@node1 for the t200d2 drive (line breaks shown here for readability):

```
# ov_dcp -T t200d2 t200d2@node1
DCPs:
Drive Name      DCP Name             DCP Disabled DCP Host     DCP Type     DCP Priority
DCP Control Path
t200d2          t200d2@node1         temporary    node1        Ultrium6     1000
/dev/ts/pci0000:0c:00.0/fc/21120090a5003194-22120090a5003194/lun0
```

   After some time (up to 6 minutes), DMF will notice this new DCP state and will shut down any mover children using this DCP.

   To permanently disable the path, use the -D option.

   Note: The behavior is different for a non-mover process, such as for dmatsnf(8), dmatread(8), or xfsdump(8). If one of these processes already has a volume mounted when the DCP / drive / library is disabled, the process will continue to completion. If using -T and one of these processes specifically asks for a mount in that drive, the mount will block. (If using -D, the mount will fail.) If a drive is disabled and one of these processes requests a mount without specifying a particular drive, it will be directed to an enabled drive in the drive group.

   ---

2. Verify that the path to the DCP is now unusable. For example, use the ov_stat(8) command to display the status of the DCPs for the t200d2 drive:

```
# ov_stat -D t200d2 -x
DCPName         DriveName Usable SoftState    Disabled  MsgLevel
t200d2@node1    t200d2    false  disconnected temporary debug
t200d2_2@node1 t200d2     true   ready        false     debug
```

To later reenable the DCP, use the `-E` option.

For more information, see the following man pages:

`ov_dcp`(8)
`ov_dumptable`(8)

Disable an OpenVault Drive

To temporarily disable one or more drives, do the following:

1. Disable the individual drives by using the `ov_drive`(8) command. where *drive_expr* may be a regular expression:

   # **ov_drive -T *drive_expr***

   For example, to disable all drives that have names that begin with `t200`:

   ```
   # ov_drive -T 't200.*'
   Drives:
   Drive           Drive Group     Disabled     Vendor Product
   Serial Number
   t200d3          drives          temporary    IBM ULTRIUM-TD5     1013003194
   t200d1          drives          temporary    IBM ULTRIUM-TD6     1011003194
   t200d2          drives          temporary    IBM ULTRIUM-TD6     1012003194
   ```

   After some time (up to 6 minutes), DMF will notice this new drive state and will shut down any mover children using these drives.

   Note: The behavior is different for a non-mover process. See the Note above in "Disable an OpenVault DCP" on page 590.

   To permanently disable a drive, use the `-D` option instead. For more details, see the `ov_drive`(8) man page.

2. Verify that the drives are disabled by using the ov_stat(8) command. To display the status of all drives, use the -d option:

   # **ov_stat -d**

   To verify specific drives, use the -D option. For example, the following output shows that all of the drives that begin with t200 are unusable and temporarily disabled:

```
# ov_stat -D 't200.*'
DriveName Group  Usable Access Disabled  SoftState HardState Occupied PCL
t200d1    drives false  true   temporary ready     unloaded  false
t200d2    drives false  true   temporary ready     unloaded  false
t200d3    drives false  true   temporary ready     unloaded  false
```

To reenable the drives, use the ov_drive -E. For more information, see the ov_drive(8) man page.

Disable an OpenVault Library

To disable the entire library, do the following:

1. Disable the library by using one of the following ov_library(8) command lines. After some time (up to 6 minutes), DMF will notice this new library state and will shut down any mover children using the drives in this library. For more details, see the ov_library(8) man page.

   Note: The behavior is different for a non-mover process. See the **Note** above in "Disable an OpenVault DCP" on page 590.

   - If you make copies of files in separate libraries, you can use the following command to disable the primary library and redirect recalls to the other library, where *lib_expr* may be a regular expression:

     # **ov_library -D *lib_expr***

     For example:

     # **ov_library -D t120**

After you use this command, the ov_stat command will report a Disabled state of permanent. So long as the other library is not also disabled with the -D option, recall requests will be forwarded to the other library. Migrate requests to the disabled library will queue. An OpenVault mount request to this library (such as if you are using xfsbackup) would fail.

- If there is not another library, use the following command.

  # **ov_library -T *lib_expr***

  For example:

  # **ov_library -T t120**

2. Verify the Disabled state of the library by using the -l option to ov_stat.For example, the following output shows that the t120 library is temporarily disabled:

```
# ov_stat -l
LibraryName Usable Disabled
t120        false  temporary
t200        true   false
```

3. Verify that all of the drives in the library have a SoftState state of ready (and are therefore currently unused) by using the ov_stat -d command. For example, the following output shows that although the drives are still enabled (with a false state for Disabled because only the library was disabled), they are unused (because the SoftState state is ready):

```
# ov_stat -d
DriveName Group  Usable Access Disabled  SoftState HardState Occupied PCL
t120d1    drives true   true   false     ready     unloaded false
t120d2    drives true   true   false     ready     unloaded false
t120d3    drives true   true   false     ready     unloaded false
```

To reenable the library, use the ov_library -E option.

For more information, see the following man pages:

- ov_library(8)
- ov_stat(8)

Disable a TMF Drive

To disable TMF drives, do the following:

1. Disable the drives by using the `tmconfig`(8) command:

   # **`tmconfig` *`drivenames`* `down`**

   For example, to disable the two drives `tape01` and `tape03`:

   # **`tmconfig tape01:tape03 down`**

   After some time (up to 6 minutes), DMF will notice this new drive state and will shut down any mover children using these drives.

   Note: The behavior is different for a non-mover process. See the Note above in "Disable an OpenVault DCP" on page 590.

   ---

2. Verify that the drives are unused and have a `stat` status of `down` by examining the output of the `tmstat`(8) command:

   # **`tmstat`**

   For example, the following output shows that both `tape01` and `tape03` are down (`down` in the `stat` field) and unused (empty `user` and `session` fields):

```
# tmstat
user  session  group  a stat device  stm rl ivsn  evsn  blocks
               dga    - down tape01
               dga    - down tape03
```

To reenable the drives, use the `tmconfig up` command.

For more information, see the following man pages:

- `tmconfig`(8)
- `tmstat`(8)

Stop the COPAN VTL

Following is one way stop COPAN VTL:

1. Disable the COPAN VTL drives gracefully. See "Temporarily Disable Components Before Maintenance" on page 128.

   Do not proceed to the next step until you verify that all of the drives are disabled.

2. Stop the LCPs associated with the COPAN drives:

   - If you have only COPAN drives, stop all of the LCPs:

     # **ov_stop lcp**

   - If you have a mix of COPAN VTLs and physical tape libraries, stop just the COPAN LCPs:

     # **ov_stop *COPAN_LCP1 COPAN_LCP2 ...***

     For example, if there are four COPAN LCPs named `C00-C03`:

     # **ov_stop C00 C01 C02 C03**

3. When you want to restart the COPAN VTL, restart the LCPs associated with the COPAN VTL:

   - If you have only COPAN drives, start all of the LCPs:

     # **ov_start lcp**

   - If you have a mix of COPAN VTLs and physical tape libraries, start just the COPAN LCPs:

     # **ov_start *COPAN_LCP1 COPAN_LCP2 ...***

     For example, if there are four COPAN LCPs named `C00-C03`:

     # **ov_start C00 C01 C02 C03**

4. Reenable the COPAN drives, where *COPAN_drive_expr* may be a regular expression:

   # **ov_drive -E *COPAN_drive_expr***

DMF SOAP Server

This chapter discusses the following:

- "Overview of DMF SOAP" on page 597
- "Accessing the DMF SOAP and WSDL" on page 599
- "Starting and Stopping the DMF SOAP Service" on page 600
- "Security / Authentication" on page 601
- "DMF SOAP Sample Client Files" on page 601

Overview of DMF SOAP

DMF provides access to the following functions via the DMF Simple Object Access Protocol (SOAP) web service:

```
dmarchive
dmattr
dmget
dmoper
dmprojid
dmput
dmrestore
dmsoapquery
dmtag
dmversion
```

Note: A limited set of options are available for these commands via DMF SOAP. For more information, click on the operation name in the SOAP interface and read the information under the **Documentation** heading displayed.

DMF SOAP log files are kept in *SPOOL_DIR*/`dmfsoap`.

Figure 16-1 shows an example of the documentation for the `ws_dmattr` operation.

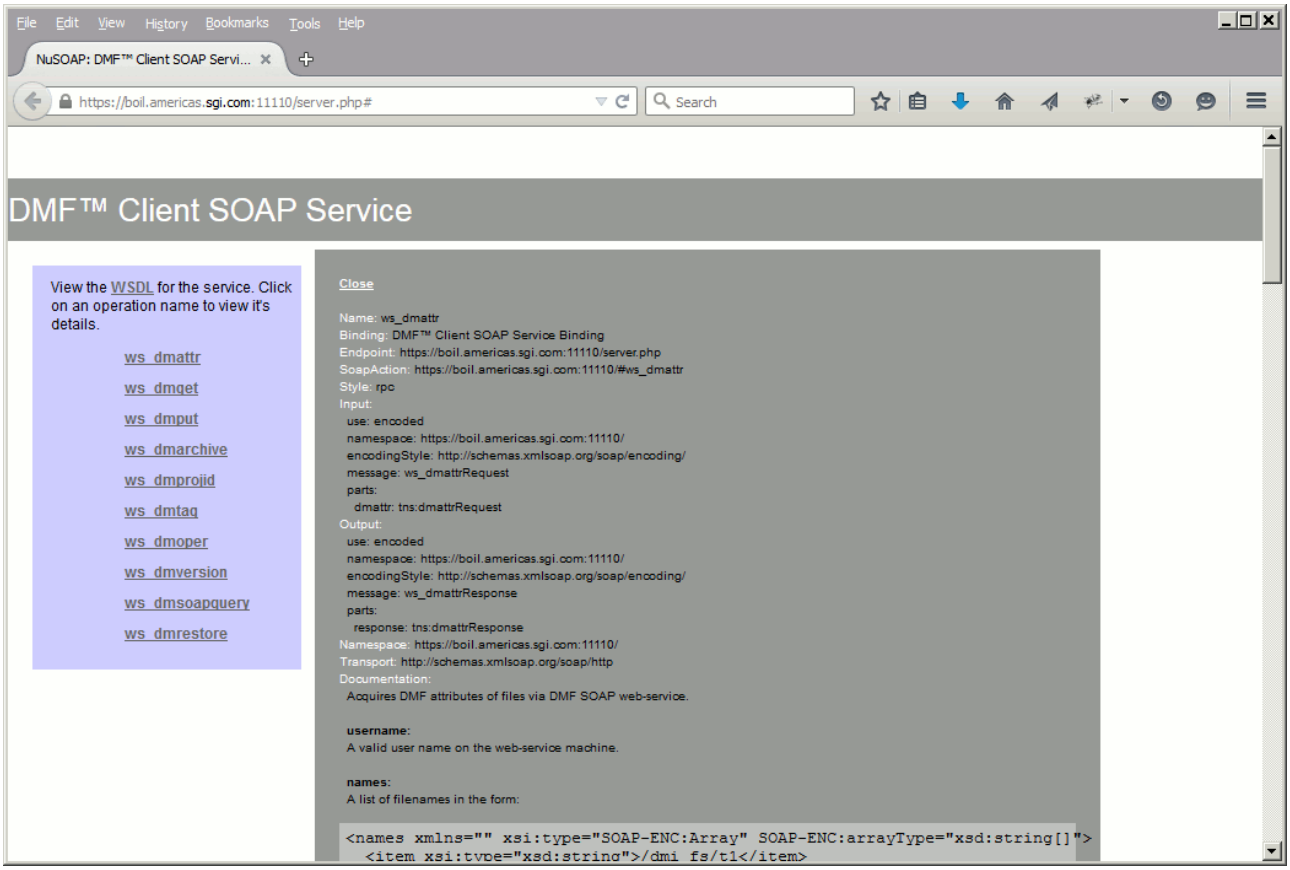

Figure 16-1 DMF SOAP Operation Documentation

Accessing the DMF SOAP and WSDL

To access DMF SOAP, do the following:

1. Point your browser to the following secure address:

   `https://`*YOUR_DMF_SERVER*`:11110/server.php`

2. Accept the security certificate.

   Note: DMF SOAP generates its own SSL certificates, rather than having the SSL certificates signed by a commercial certificate authority. Therefore, the certificate warning is safe to ignore.

3. Enter the DMF SOAP service user name (`dmfsoap`) and password. The default password is `INSECURE`.

   Note: To change the password to something site-specific (*NEWPASSWORD*), run the following command:

```
# htpasswd2 -b -c /usr/share/dmfsoap/passwords/passwds dmfsoap NEWPASSWORD
```

4. To access the web service definition language (WSDL) definition, click on **WSDL** in the interface. Figure 16-2 shows an example. Use the brower's **Save As...** feature to save the WSDL to a file for consumption.

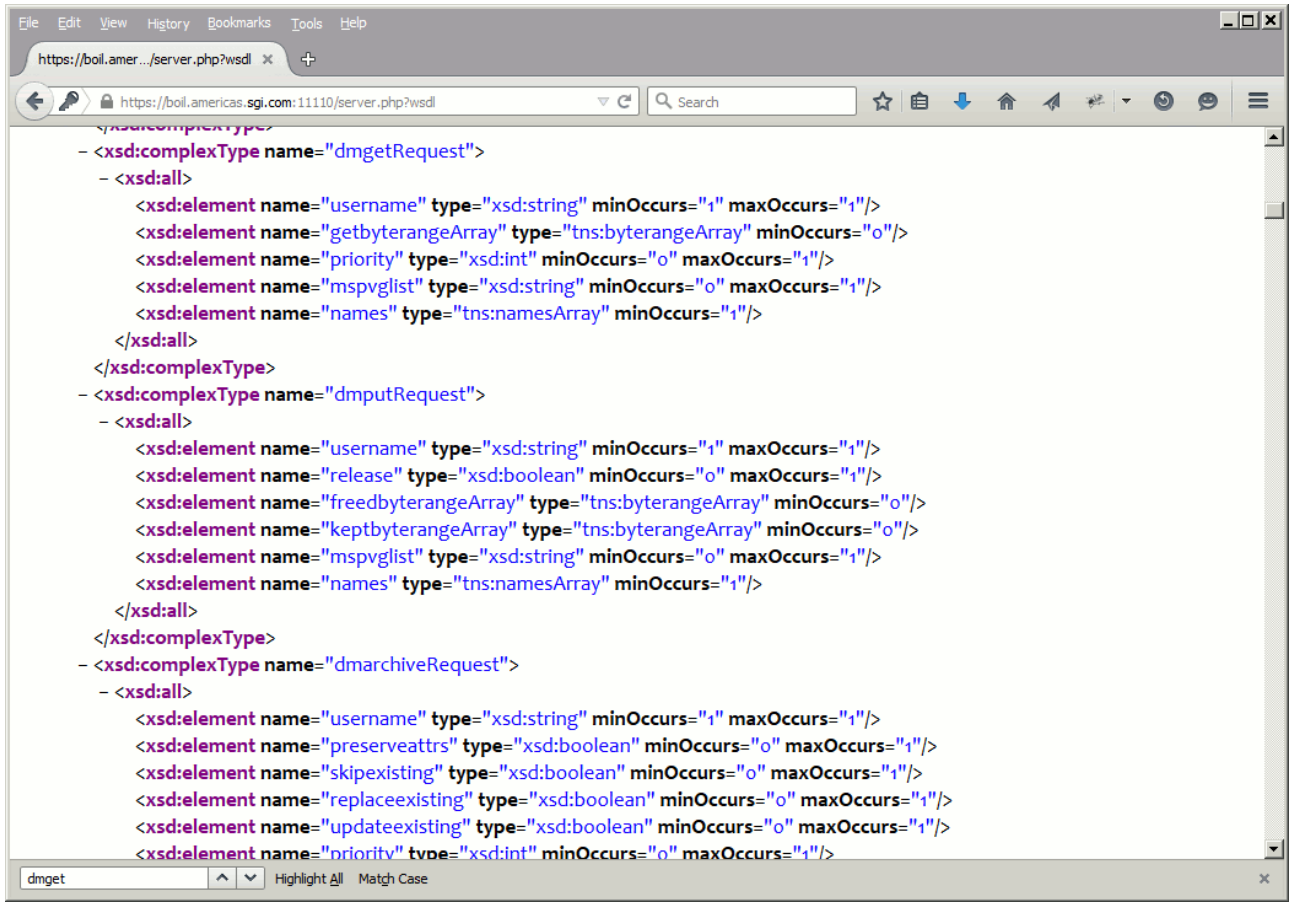

Figure 16-2 WSDL Definition

Starting and Stopping the DMF SOAP Service

This section discusses the following:

- "Starting the `dmfsoap` Service" on page 601
- "Preventing Automatic Start of `dmfsoap` After Reboot" on page 601
- "Explicitly Stopping `dmfsoap`" on page 601

Starting the dmfsoap Service

The dmfsoap service for DMF SOAP is off by default.

To start the service explicitly, execute the following on the DMF server:

```
dmfserver# service dmfsoap start
```

Preventing Automatic Start of dmfsoap After Reboot

To prevent automatic startup of the DMF environment, execute the following chkconfig(8) commands as root on the DMF server:

```
dmfserver# chkconfig dmfsoap off
```

Explicitly Stopping dmfsoap

To stop the DMF environment daemons explicitly, execute the following on the DMF server:

```
dmfserver# service dmfsoap stop
```

Security/Authentication

DMF SOAP uses basic access authentication, via HTTPS, when making a request. Encapsulated in the request is the user name on the DMF system. DMF SOAP checks that the user name supplied is a valid and executes the DMF command as that user. However, no additional authentication is done; the client has complete responsibility for user authentication.

DMF SOAP Sample Client Files

DMF provides the following sample PHP files via the web interface that you can use to remotely access the DMF SOAP service:

```
dmarchive.php
dmattr.php
dmget.php
dmoper.php
```

```
dmprojid.php
dmput.php
dmrestore.php
dmsoapquery.php
dmtag.php
dmversion.php
```

These files are installed in the following directory:

/usr/share/doc/dmf-*VERSION*/info/sample/dmfsoap_client

These files are simply for demonstration purposes. You can copy them the remote machine from which you want to access the DMF SOAP service and modify as needed.

Caution: There is no security provided by these sample files.

To use the sample files, you must install the following packages:

```
php5-soap
php-curl
php5-openssl
```

For more information, see the README file and the comments in the .php files.

For example, the following is the sample file for dmget.php (line breaks added here for readability):

```
<?php
        #
        # Copyright 2012 SGI.   All rights reserved.
        #
        # This is a simple example SOAP Client interface to the DMF SOAP
        # Service supporting the dmget interface.
        # Note: You may need to install the php curl and php openssl
        # packages on your machine.

        # Nothing SOAP related yet.   Just set up some variables specific
        # to dmget and parse the command line arguments.
        $priority = null;
        $getByteRangeArray = array();
        $getbyterange_cnt = 0;
        $getbyterange = false;
```

```
        $request = array();
        $namesArray = array();
        for( $i = 0; $i < $argc; $i++ ) {
                if( $argv[$i] == '-h' ) {
                        echo "Usage: php dmget.php WSDL login password username
[-U priority] [-B getbyterange-list] fullpathnametofile1 fullpathnametofile2 \nWhere
getbyterange-list can be a comma separated list (no spaces)\nExamples:\n\tphp dmget.php
https://machine.com:11110/server.php?wsdl dmfsoap INSECURE username
-B 0:4000 /dmf_fs/testfile\n";
                        exit;
                }
                if( $argv[$i] == '-B' ) {
                        $i++;
                        $getbyterange = true;
                        $values_array = explode(",", $argv[$i]);
                        foreach($values_array as $value) {
                                $value_array = explode(":", $value);
                                $start = $value_array[0];
                                $end = $value_array[1];
                                $getByteRangeArray[$getbyterange_cnt] = array('start' =>
$start, 'end' => $end);$getbyterange_cnt = $getbyterange_cnt + 1;
                        }
                        continue;
                }
                if( $argv[$i] == '-U' ) {
                                $i++;
                                $priority = $argv[$i];
                                continue;
                }
                if( $i < 5 ) {
                                continue;
                }
                array_push($namesArray, $argv[$i]);
        }

        # Here's the SOAP work
        $wsdl = $argv[1];
        $login = $argv[2];
        $password = $argv[3];
        $username= $argv[4];
        $request['username']=$username;
```

```
        $request['getbyterangeArray']=$getByteRangeArray;
        $request['priority']=$priority;
        $request['names']=$namesArray;

        # First, initialize the connection.
        $client = new SoapClient($wsdl, array('login'=>$login, 'password'=>$password));

        # Now make the request
        try {
                $result = $client->ws_dmget($request);
        } catch( SoapFault $fault ) {
                print("Sorry, $argv[0] returned the following ERROR: ".$fault->faultcode."-".$fault-faultstring."\n")
                return;
        }


        # Finally print the results
        print_r($result);
?>
```

DMF Queue Viewing

This section discusses the following:

- "QView Tab and the `dmqview` Command" on page 605
- "Taking a Snapshot of the DMF Requests" on page 606
- "Queue Reports" on page 606
- "Canceling a Request" on page 613
- "Reprioritizing a Request" on page 615
- "Modifying the Report Output" on page 615
- "Example of Queue-Related Tasks" on page 619
- "Database Tables and Fields" on page 620

QView Tab and the `dmqview` Command

The **QView** tab of DMF Manager and the `dmqview`(8) command display information about the internal DMF processing queues. You must log into DMF Manager to use **QView** or be `root` to run `dmqview`.

The reports that the tools display contain details about how requests relate to each other, which you can use to answer questions such as the following:

- How many recalls are in the queue?
- Where is the request for a specific file in the queue, relative to other requests?
- What is the ID of a given request, so that I can cancel or reprioritize it?

The **QView** tab in DMF Manager also lets you directly cancel and reprioritize requests; the `dmcancel`(8) and `dmrepri`(8) commands provide the same functionality for the command line.

Taking a Snapshot of the DMF Requests

To view queue information, you must first take a snapshot of the current DMF requests by using the `dmqview --snap` option or by using the following **QView** menu selection in DMF Manager:

QView
> Refresh QView

This action pauses DMF activity in order to create several relational-database tables (see "PostgreSQL Database Tables" on page 621) that reflect all of the current DMF requests. There may be hundreds of thousands of requests in the queue. The snapshot normally completes in a few seconds, but may take longer if a daemon is working on a time-consuming operation and does not immediately notice the snapshot request. The snapshot represents a static view of the requests and therefore will become stale over time. However, because it momentarily halts all activity, you should refresh the snapshot only when necessary.

Queue Reports

After a snapshot exists, you can issue a series of queries against that snapshot by selecting various reports:

- "Summary Report and Corresponding State Reports" on page 606
- "Composite Reports" on page 609

Note: These queries have no affect on the daemon, but they can impact performance if they require joining multiple database tables in order to access the required data. If a refresh is required, you can select the **Refresh QView** menu again.

Summary Report and Corresponding State Reports

The **Summary** report displays a brief listing of all request states, as supplied by the **Chunk** (`chunk`) and **Request** (`dmn_request`) database tables, see "PostgreSQL Database Tables" on page 621.

Each item reported by a bar graph at the top of the page has a corresponding detailed report listed under the **Summary** heading. Figure 17-1 shows an example. You can also click on a bar in the graph to display that state report.

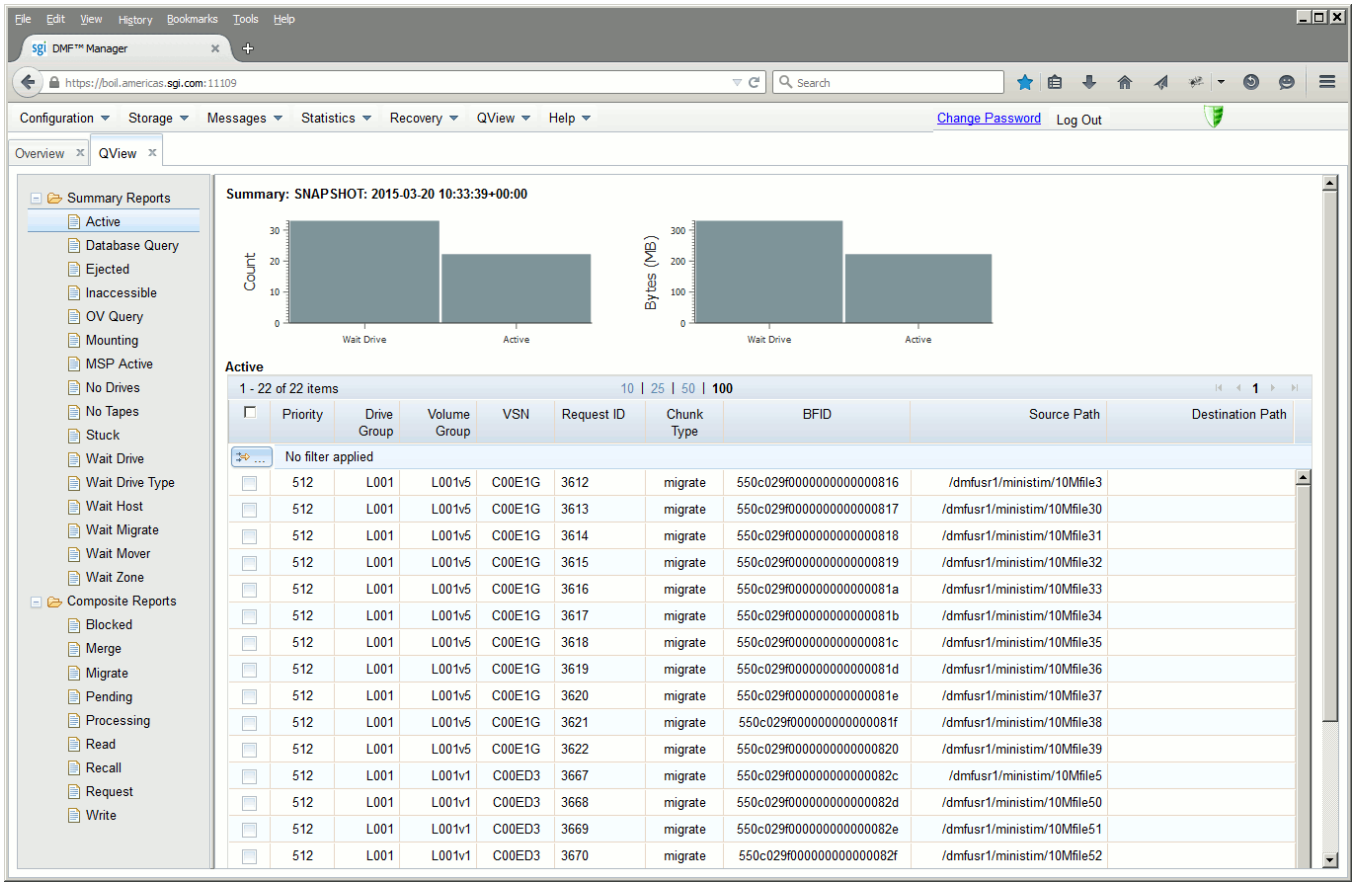

Figure 17-1 QView: Summary Reports

The summary report is available at the command line with the `--report summary` option:

```
# dmqview --report summary
        REPORT summary:    SNAPSHOT: 2015-03-20 10:33:39+00:00
          Status           Count               Bytes
     wait_drive              33           330005736
          active             22           220003828
```

The report names used by the command line are lowercase and one string (such as `--report wait_host`). See the `dmqview`(8) man page for details.

All reports other than **Active** and **MSP Active** indicate that the requests are waiting to be processed. Some reports are specific to the library server (LS), a media-specific process (MSP), or a volume group (VG).

Table 17-1 Summary Reports

Report Name	Description
Active	Displays the requests that are being worked on by a data-mover process of the LS.
Database Query	Displays the requests that are waiting because a database query is being performed.
Ejected	Displays the requests that are waiting because the physical tape has been ejected.
Inaccessible	Displays the requests that are waiting because a volume cannot be mounted. This status usually means that there is an issue with the physical library.
Mounting	Displays the requests that are waiting for the required volume to be mounted.
MSP Active	Displays the requests that are being worked on by an MSP.
No Drives	Displays the requests that are waiting because there are no usable drives. (For example, all of the drives may be down.)
No Tapes	Displays the requests that are waiting because there are no usable volumes.
OV Query	Displays the requests that are waiting for the LS to query OpenVault.
Stuck	Displays the requests that are waiting because of a physical tape that is stuck in the library.

Report Name	Description
Wait Drive	Displays the requests that are waiting for a drive. (Usable drives exist, but are unavailable. A drive may be unavailable for a variety of reasons, such as because it is handling another request or because the VG is already using the maximum number of drives allowed.)
Wait Drive Type	Displays the requests that are waiting for a drive of the correct type. (Other drives may be available, but are inappropriate.)
Wait Host	Displays the requests that are waiting for the host on which the volume will be mounted.
Wait Migrate	Displays socket merge requests that are waiting for a write child to be started.
Wait Mover	Displays the requests that are waiting for a data-mover process.
Wait Zone	Displays the requests that are waiting because there is not yet enough migrate data queued to start a mover process.

Composite Reports

The reports listed in Table 17-2 represent states that are compiled by examining multiple tables, rather than states literally reflected in the **Chunk** (`chunk`) and **Request** (`dmn_request`) database tables, as in the detailed summary reports above. Figure 17-2 shows an example **Processing** report.

Table 17-2 Composite Reports

Report Name	Description
Blocked	Displays those requests that are currently in the queue but are stopped because an earlier request for the same file is being processed. (A request will be displayed by either the **Processing** report or the **Blocked** report.)
Merge	Displays those requests that are currently merging data.

Report Name	Description
Migrate	Displays those requests that are in the process of migrating, both explicit requests submitted via the `dmput` command and requests submitted to `dmfdaemon` in order to free filesystem space as part of automated space management. The **Migrate** report is a subset of the **Processing** report.
Pending	Displays those requests that are in the queue but are waiting to become active. This includes requests that are waiting for a data-mover process, a drive, or a zone to become available. In most cases, the delay is temporary.
Processing	Displays all requests that are working or are waiting for any reason, other than stopped due to a prior request for the same file. (A request will be displayed by either the **Processing** report or the **Blocked** report.)
Read	Displays all requests that are processing and currently either reading or queued to read data from LS volumes. Merges are not included in this report.
Recall	Displays recall requests from explicit `dmget`, `dmunput`, and `dmmove` commands and kernel `krclrea` requests to read an offline file. The **Recall** report is a subset of the **Processing** report.

Report Name	Description
Request	Display all requests to `dmfdaemon` that have been explicitly requested via the command line or by the kernel. (However, merges do not appear in this report.) The request types are as follows: • `archive` puts a file under DMF management by using the `dmarchive` command • `backup` migrates a file by using the `dmput`, `dmmigrate`, or `dmfsfree` command • `copy` copies a portion of a migrated file to disk by using the `dmcopy` or `dmarchive` command • `krclrea` recalls an offline file for reading, performed by the kernel • `krclvoi` recalls of an offline file prior to overwriting or truncating, performed by the kernel • `move` moves a migrated file to new MSP or VG by using the `dmmove` or `dmdskfree` command • `recall` requests that a file or portion of a file be retrieved by using the `dmget` command • `usrvoid` retrieves a file and remove it from DMF management by using the `dmunput` command
Write	Displays all requests for which data-mover processes are currently writing data to volumes, which can be from migrates or merges issued by `dmfdaemon` or by explicit `dmdskfree` or `dmmove` commands. The **Write** report is a subset of the **Migrate** report.

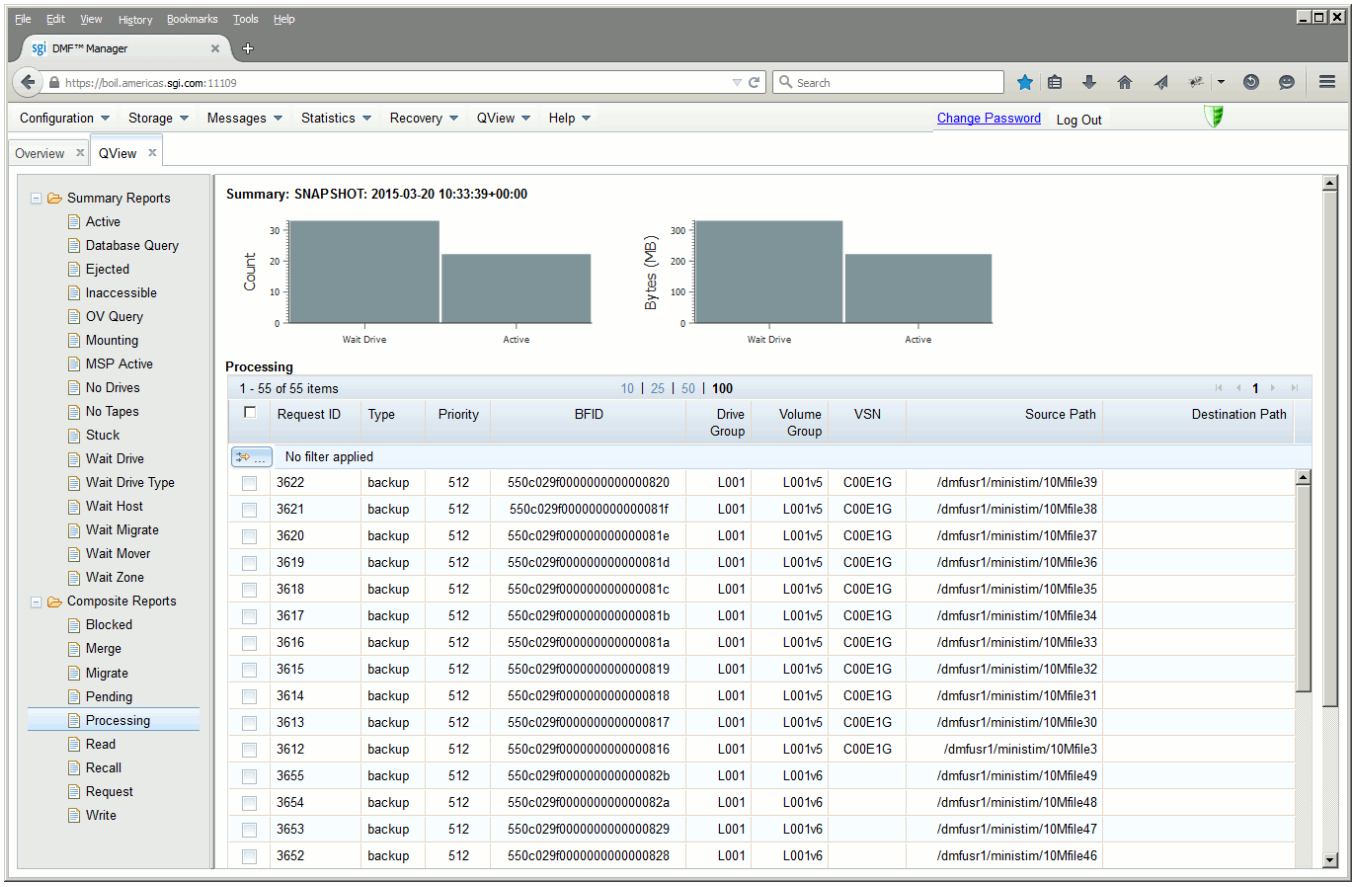

Figure 17-2 Processing Report

The report names used by the command line are lowercase (such as `--report processing`). See the `dmqview`(8) man page for details.

Canceling a Request

To cancel a request, select it and choose **Cancel Request**, as shown in Figure 17-3.

Note: Canceling a request refreshes the snapshot. Canceling can also have consequences, such as I/O failures in the application that generated the request. For more information, see the `dmcancel`(8) man page.

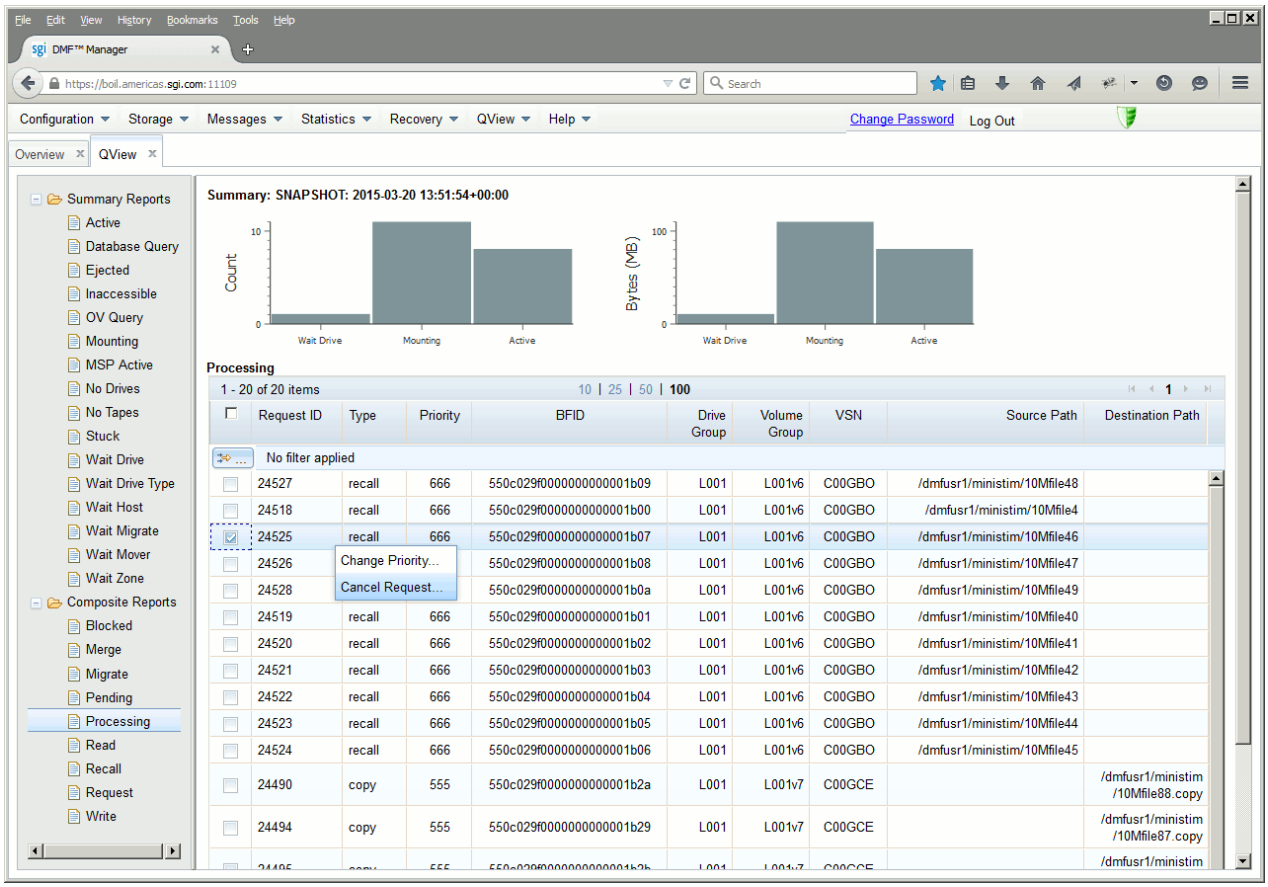

Figure 17-3 Canceling a Request

To cancel a request via the command line, view one of the reports (such as `processing`) to determine the request ID and the `dmfdaemon` PID, and use those as input to the `dmcancel` command. For example, to cancel the processing of the request for `/dmfusr1/ministim/10Mfile46`:

Note: The examples in this chapter have been adjusted for purposes of display, such as compression of white space and truncation or omission of fields.

```
# dmqview -r processing
  REPORT: processing   DMF Daemon PID: 19352  SNAPSHOT: 2015-03-20 13:51:54+00:00
Line     ReqID      Type                        BFID  Priority        VG       VSN                 Source Path
   1     24527    recall   550c029f0000000000001b09       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile48
   2     24518    recall   550c029f0000000000001b00       666    L001v6    C00GBO  /dmfusr1/ministim/10Mfile4
   3     24525    recall   550c029f0000000000001b07       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile46
   4     24526    recall   550c029f0000000000001b08       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile47
   5     24528    recall   550c029f0000000000001b0a       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile49
...
# dmcancel -p 19352 24525
```

For more information, see the `dmcancel`(8) man page.

Reprioritizing a Request

To reprioritize a request, select it and choose **Change Priority**.

Note: Reprioritizing a request refreshes the snapshot.

To reprioritize a request via the command line, view one of the reports (such as `processing`) to determine the request ID and the `dmfdaemon` PID, and use those as input to the `dmrepri` command. For example, to reprioritize the processing of file `/dmfusr1/ministim/10Mfile49` so that it has the highest priority:

```
# dmqview -r processing
  REPORT: processing   DMF Daemon PID: 19352  SNAPSHOT: 2015-03-20 13:51:54+00:00
Line     ReqID      Type                        BFID  Priority        VG       VSN                 Source Path
   1     24527    recall   550c029f0000000000001b09       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile48
   2     24518    recall   550c029f0000000000001b00       666    L001v6    C00GBO  /dmfusr1/ministim/10Mfile4
   3     24525    recall   550c029f0000000000001b07       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile46
   4     24526    recall   550c029f0000000000001b08       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile47
   5     24528    recall   550c029f0000000000001b0a       666    L001v6    C00GBO /dmfusr1/ministim/10Mfile49
...
 # dmrepri -p 19352 -U 1024 24528
```

For more information, see the `dmrepri`(8) man page.

Modifying the Report Output

This section discusses the following:

- "Modifying the QView Display" on page 615
- "Modifying the `dmqview` Output" on page 617

Modifying the QView Display

The reports are based on information gathered from various database tables and fields. You can modify the output of these reports by adding or subtracting the fields shown by selecting the following:

QView> Show QView Fields

For example, Figure 17-4 shows the fields available for the **Active** report. Those fields that are checked are shown by default; to modify the report, you can select additional fields or deselect default fields. (For more information about the fields, see "Database Tables and Fields" on page 620.)

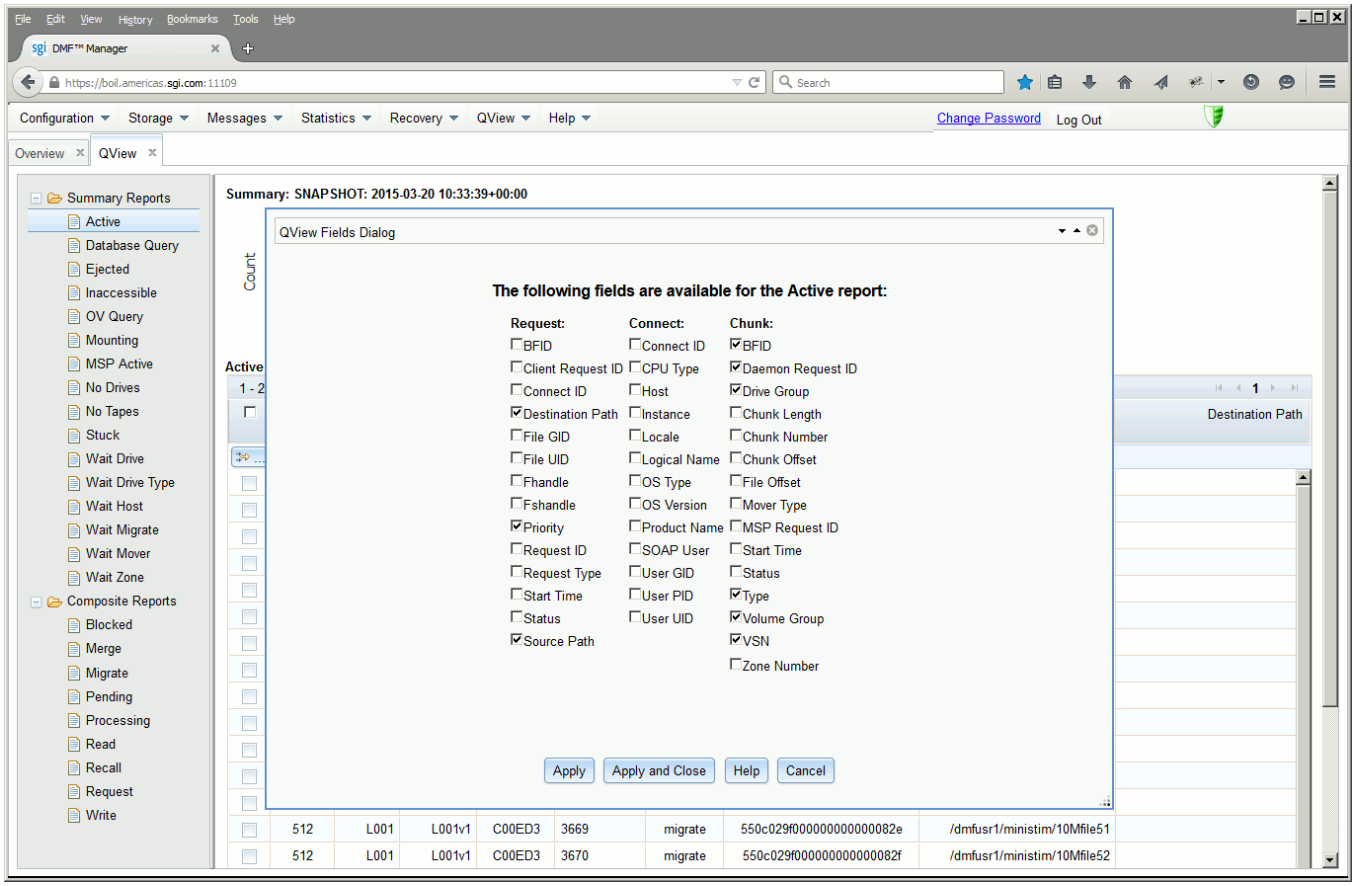

Figure 17-4 Fields Available for Display by a QView **Active** Report

Different reports use different database tables. For example, the above shows that the **Active** report uses the **Request** (`dmn_request`), **Connect** (`dmn_connect`), and **Chunk** (`chunk`) tables:

- To remove the **Priority** column from the report, deselect the **Priority** field under **Request**.
- To add the **Host** column, select the **Host** field under **Connect**.
- To apply your changes, click **Apply and Close**.

For details about the available reports, see "Queue Reports" on page 606.

For details about the individual fields, see "Database Tables and Fields" on page 620. This information is also available from the **Help** button on the **QView Fields Dialog** page and from the **What is ...** selection for each column header in the report.

You can also customize the graphical display by using filters, similar to "Filtering Alerts" on page 225.

Modifying the `dmqview` Output

With the command line, you can use the `--addfields` option to add the specified options or the `--onlyfields` option to restrict the output to only the specified fields. To view all available fields for a given report, use the `--fields` option. (For more details, see "Database Tables and Fields" on page 620.)

You can further control the display with the `--whereorders` option, which lets you modify the PostgreSQL database query. You can include multiple conditions such as `=`, `AND`, `OR`, and `LIKE` (case-insensitive).

For example, the following displays all of the processing requests:

```
# dmqview --report processing
  REPORT: processing   DMF Daemon PID: 19352  SNAPSHOT: 2015-03-20 09:16:32+00:00
Line     ReqID      Type                         BFID  Priority        VG       VSN                 Source Path
   1      1258    backup   550c029f0000000000000657       512    L001v3            /dmfusr1/ministim/10Mfile1
   2      1259    backup   550c029f0000000000000658       512    L001v3           /dmfusr1/ministim/10Mfile10
   3      1274    backup   550c029f0000000000000662       512    L001v5    C00E1G  /dmfusr1/ministim/10Mfile3
   4      1275    backup   550c029f0000000000000663       512    L001v5    C00E1G /dmfusr1/ministim/10Mfile30
   5      1287    backup   550c029f000000000000066f       512    L001v6    C00E26 /dmfusr1/ministim/10Mfile41
   6      1289    backup   550c029f0000000000000671       512    L001v6    C00E26 /dmfusr1/ministim/10Mfile43
   7      1334    backup   550c029f0000000000000699       512    L001v8    C00EBN  /dmfusr1/ministim/10Mfile9
   8      1335    backup   550c029f000000000000069a       512    L001v8    C00EBN /dmfusr1/ministim/10Mfile90
```

The following displays those requests in VG `L001v5` or VG `L001v8`:

```
# dmqview --report processing --whereorders "(msp_request.vg='L001v5') OR (msp_request.vg='L001v8')"
  REPORT: processing   DMF Daemon PID: 19352  SNAPSHOT: 2015-03-20 09:16:32+00:00
Line     ReqID      Type                          BFID  Priority        VG       VSN                 Source Path
   1      1274    backup   550c029f0000000000000662       512    L001v5    C00E1G  /dmfusr1/ministim/10Mfile3
   2      1275    backup   550c029f0000000000000663       512    L001v5    C00E1G /dmfusr1/ministim/10Mfile30
   3      1334    backup   550c029f0000000000000699       512    L001v8    C00EBN  /dmfusr1/ministim/10Mfile9
   4      1335    backup   550c029f000000000000069a       512    L001v8    C00EBN /dmfusr1/ministim/10Mfile90
```

The following displays the requests in LS `L00` (BFID output truncated here for purposes of display):

```
# dmqview --report processing --addfields msp_request.ls --whereorders "(msp_request.ls='L00')"
  REPORT: processing   DMF Daemon PID: 19352  SNAPSHOT: 2015-03-20 09:16:32+00:00
Line     ReqID      Type  ...BFID  Priority       VG     VSN                 Source Path  LS
   1      1258    backup  5...657       512   L001v3          /dmfusr1/ministim/10Mfile1  L00
   2      1259    backup  5...658       512   L001v3         /dmfusr1/ministim/10Mfile10  L00
   3      1274    backup  5...662       512   L001v5  C00E1G  /dmfusr1/ministim/10Mfile3  L00
   4      1275    backup  5...663       512   L001v5  C00E1G /dmfusr1/ministim/10Mfile30  L00
   5      1287    backup  5...66f       512   L001v6  C00E26 /dmfusr1/ministim/10Mfile41  L00
   6      1289    backup  5...671       512   L001v6  C00E26 /dmfusr1/ministim/10Mfile43  L00
   7      1334    backup  5...699       512   L001v8  C00EBN  /dmfusr1/ministim/10Mfile9  L00
   8      1335    backup  5...69a       512   L001v8  C00EBN /dmfusr1/ministim/10Mfile90  L00
```

Note: To act on a given field, the field must be present in the report by default or else supplied with the `--addfields` or `--onlyfields` options. Therefore, the above command line adds the `msp_request.ls` field, because it does not appear by default for the `processing` report).

For more information, see the `psql`(1) man page

Example of Queue-Related Tasks

Note: The output spacing is adjusted and some fields are omitted for purposes of readability in this document.

Suppose that user Jane issued a `dmget` of her offline (`OFL`) file `/dmfusr1/jane/janefile` and after waiting some time asks the administrator why it has not been recalled yet. You as an administrator could do the following to determine if there is a problem:

1. Log in as `root`.
2. If there is no recent queue view snapshot, obtain one:

   # **dmqview --snap**

3. Display the entire list of requests:

```
# dmqview --report request
 REPORT: request   DMF Daemon PID: 8307  SNAPSHOT: 2015-03-25 17:04:10+00:00
Line ReqID    Type                     BFID Priority                     Source Path
   1    32  backup 4d9dd5130000000000011a1a7      512  /dmfusr1/stim/operator/filevvpl
   2    34 krclrea 4d9dd5130000000000003ce53      512
   3    35 krclrea 4d9dd5130000000000034770      512
   4    36 krclrea 4d9dd5130000000000033ae5      512
   5    38  recall 4d9dd5130000000000011a1a3      512            /dmfusr1/bob/bobfile
   6    39  recall 4d9dd5130000000000011a1a5      512           /dmfusr1/bob/bobfile2
   7    40  recall 4d9dd5130000000000011a1a6      512          /dmfusr1/jane/janefile
```

Note: This step is not required, but is included for completeness for this example.

4. Look for Jane's file in the list of requests:

```
# dmqview --report recall --whereorders "(dmn_request_src_path like '%janefile%')"
 REPORT: recall   DMF Daemon PID: 1346  SNAPSHOT: 2015-03-25 15:44:13+00:00
Line ReqID    Type                     BFID Priority    VG    VSN            Source Path
   1   439  recall 4d9dd5130000000000011a1a6      512 vgc07 C07KC0 /dmfusr1/jane/janefile
```

From the above, you can see that Jane's request will use the volume with volume serial number (VSN) `C07KC0`, from VG `vgc07`. This indicates that the request has been sent to a VG.

5. Because the request has been sent to a VG, you can use the `read` report to determine its current status:

```
# dmqview --report read
 REPORT: read   DMF Daemon PID: 8307  SNAPSHOT: 2015-03-25 17:04:10+00:00
Line ReqID    Type ...    VG    VSN      Status            Source Path
   1    38  recall ... vgc07 C07KC0  wait_drive   /dmfusr1/bob/bobfile
   2    39  recall ... vgc07 C07KC0  wait_drive  /dmfusr1/bob/bobfile2
   3    40  recall ... vgc07 C07KC0  wait_drive /dmfusr1/jane/janefile
   4    34 krclrea ... vgc07 C07FB5  active
   5    35 krclrea ... vgc07 C07CBS  active
   6    36 krclrea ... vgc07 C07DAD  active
```

The above shows that Jane's request has a status of `wait_drive` (which means that it is waiting for a drive to become available) and that there are two other `recall` requests (`/dmfusr1/bob/bobfile` and `/dmfusr1/bob/bobfile`) that will use the same volume.

6. To determine the order in which the requests will be recalled when the volume is mounted, you could use the following query:

```
#  dmqview --report read --addfields "chunk.chunk_number,chunk.zone_number,chunk.chunk_length" \
--whereorders "(chunk.vsn='C07KC0')" --sortby chunk.chunk_number  --sortorder ASC
 REPORT: read   DMF Daemon PID: 8307  SNAPSHOT: 2015-03-25 17:04:10+00:00
Line ReqID    Type ...    VG    VSN     Status            Source Path ... Chunk Number Zone Number Chunk Length
   1    38  recall ... vgc07 C07KC0 wait_drive   /dmfusr1/bob/bobfile              155          10     70000000
   2    39  recall ... vgc07 C07KC0 wait_drive  /dmfusr1/bob/bobfile2              159          12   2000000000
   3    40  recall ... vgc07 C07KC0 wait_drive /dmfusr1/jane/janefile              161          13     80000000
```

Recall requests on a tape are processed in tape order (starting at the current tape position if the tape is already mounted and recalling files). Tape order is indicated by the chunk number. (If the tape in question was already mounted and doing work, the `Status` output would be `active` and you could use the `dmstat -t` option to determine the zone that is currently being used; zones are processed in tape order.)

Database Tables and Fields

This section discusses the following:

- "PostgreSQL Database Tables" on page 621

- "Fields for the Chunk Database Table" on page 624
- "Fields for the Connect Database Table" on page 625
- "Fields for the Filesys Database Table" on page 627
- "Fields for the Daemon Request Database Table" on page 626
- "Fields for the Mover Database Table" on page 628
- "Fields for the MSP Request Database Table" on page 629

The table and field names used by the command line are lowercase and a single string and in some cases abbreviated, in the following format:

tablename.*fieldname*

For example, for the daemon request table and destination path:

`dmn_request.dst_path`

See the `dmqview`(8) man page for details.

PostgreSQL Database Tables

The **QView** reports obtain information from several relational-database tables, depending upon the selected report. Each table has many fields; some fields may appear in multiple tables. Table 17-3 summarizes the database tables.

Table 17-3 PostgreSQL Database Tables

PostgreSQL Table	DMF Manager Name	Description
`chunk`	**Chunk**	Contains one record for each data chunk used for a media-specific process (MSP) or VG request. (Most small files are written as single chunks. When a migrated file cannot fit onto a single volume, the file is split into multiple chunks.) See "Fields for the Chunk Database Table" on page 624.
`dmn_connect`	**Connect**	Contains one record for each DMF client connection to `dmfdaemon`, specifying the node and program that made the connection. See "Fields for the Connect Database Table" on page 625.
`dmn_request`	**Request**	Contains one record for each request to `dmfdaemon` that involves data movement. (Multiple `dmn_request` records may reference a single `dmn_connect` record, but it is possible for a `dmn_connect` record to have no corresponding `dmn_request` record if none of the requests associated with that connection are related to data transfer.) See "Fields for the Daemon Request Database Table" on page 626.
`filesys`	**Filesys**	Contains a record for each filesystem managed by DMF. See "Fields for the Filesys Database Table" on page 627.
`globals`	—	Contains the `dmfdaemon` process ID, which appears in the header of each report.

PostgreSQL Table	DMF Manager Name	Description
`mover`	**Mover**	Contains one record for each read or write data-mover process within a VG. Many chunk records may point to a single `mover` record. See "Fields for the Mover Database Table" on page 628.
`msp_request`	**MSP Request**	Contains one record for each request sent to an MSP/VG. There may be multiple `msp_request` records associated with a single `dmn_request` record. There may be multiple `chunk` records associated with a single `msp_request` record. See "Fields for the MSP Request Database Table" on page 629.

Fields for the Chunk Database Table

Table 17-4 describes the fields available for the **Chunk** (`chunk`) database table.

Table 17-4 Chunk Fields

Field	Description
BFID	The bit-file identifier (BFID) assigned to each file during the migration process, which links a migrated file to its data on alternate media.
Chunk Length	The length of the chunk, in bytes.
Chunk Offset	The byte location in the source file where the data transfer will begin.
Drive Group	Name of the DMF drive group that contains the chunk.
Daemon Request ID	The ID associated with this chunk by `dmfdaemon`.
File Offset	Byte offset within the destination file copy where the transfer is to begin. This value will match the **Chunk Offset** value for all request types except for a request issued by a `dmcopy` command.
MSP Request ID	A number that maps this record in the `chunk` table to the record in the `msp_request` table.
Start Time	The time at which the request was received by the VG.
Status	The status of the request. A value of active indicates that the chunk is being processed by a data-mover process; any other value indicates why the request is not currently being processed. These states are reflected in the**Summary** and its corresponding detailed reports.
Mover Type	The type of data-mover process that will act on this request (either `write` or `read`).
Type	The type of the chunk action (one of `merge`, `migrate`, or `recall`).
Volume Group	The name of the VG that contains this chunk.

Field	Description
VSN	The volume serial number (VSN) of the volume that contains this chunk. For chunks with a type of `migrate`, this field will be null until a data-mover process is forked to perform the migration.
Zone Number	The zone number within the volume (specified by the **VSN** field) where the chunk resides.

Fields for the Connect Database Table

Table 17-5 describes the fields for the **Connect** (`dmn_connect`) database table.

Table 17-5 Connect Fields

Field	Description
Connect ID	The connection ID number used to map `dmn_connect` and `dmn_request` records to the connection by which they arrived.
CPU Type	The CPU architecture of the host that is running the program that created this connection (specified by the **Logical Name** field.
Host	The hostname where the program is running.
Instance	The instance ID of the program. (Normally, a process will only maintain one connection instance to `dmfdaemon`.)
Locale	The language being used by the program, as output by the Linux `locale`(1) command (such as `en_US.UTF-8`).
Logical Name	The program that created this connection to `dmfdaemon` (such as `/usr/bin/dmget`).
OS Type	The operating system on the host where the program is running.
OS Version	The operating system version on the host where the program is running.

Field	Description
Product Name	The name and version of the software product of which the program is a member.
SOAP User	True (`t`) if this connection was made using SOAP; in this case, the **User UID** and **User GID** will be the user ID and group ID of the user who executed the SOAP request. If SOAP was not used, this value is false (`f`).
User GID	The group ID of the process that initiated the program.
User PID	The process ID of the process that initiated the program.
User UID	The user ID of the process that initiated the program.

Fields for the Daemon Request Database Table

Table 17-6 describes the fields available for the **Daemon Request** (`dmn_request`) database table.

Table 17-6 Daemon Request Fields

Field	Description
BFID	The BFID for the requested file.
Client Request ID	The client's request ID if this request arrived via a client connection (this field will be null for kernel requests).
Connect ID	A number that maps this record in the `dmn_connect` table to the record in the `dmn_request` table.
Destination Path	The destination path where the file is being transferred. This will be null except for copy and archive requests.
File GID	The group ID of the owner of the file in the DMF-managed filesystem. (This field may be null depending upon the current state of the request.)
File UID	The user ID of the owner of the file in the DMF-managed filesystem. (This field may be null depending upon the current state of the request.)

Field	Description
Fhandle	Handle for the file on which this request is based.
Fshandle	Handle for the filesystem in which the file resides.
Priority	Priority of the request, in the range `1` (lowest priority) through `1024` (highest priority). The default priority is `512`.
Request ID	The daemon request ID.
Request Type	The daemon request type.
Source Path	The pathname of the file that is the basis of the request. (Note, this will be null for kernel `krclrea` and `krclvoi` requests because pathnames are not available from the kernel.)
Start Time	The time at which this request was received by `dmfdaemon`.
Status	The status of this request, either `active` (displayed by the **Processing** report) or `blocked` (displayed by the **Blocked** report)

Fields for the Filesys Database Table

Table 17-7 describes the fields available for the **Filesys** (`filesys`) database table.

Table 17-7 Filesys Fields

Field	Description
Device	The device number (`dev_t`) for the filesystem.
Device Name	The device pathname for the filesystem. This could be in any of the following forms: `/dev/lxvm/`*name* `/dev/cxvm/`*name* `/dev/xvm-`*N*`/`*name*
Directory	The mount point for the filesystem.
Fshandle	The fhandle for the filesystem.
FS Type	The filesystem type.

Fields for the Mover Database Table

Table 17-8 describes the fields available for the **Mover** (`mover`) database table.

Table 17-8 Mover Fields

Field	Description
Drive Group	The DMF drive group that contains the volume.
Drive	The drive used for the mount. This will be null for data-mover processes with a status other than `active`.
Host	The host on which the volume will be mounted. This will be null for data-mover processes with a status other than `active`.
Position	For data-mover processes in `pending` status, this is the relative position of the mover in the queue. For other status types, this field will be null.
Priority	Priority of the highest-priority request assigned to the data-mover process for the volume, in the range `1` (lowest priority) through `1024` (highest priority). The default priority is `512`. (If a data-mover process is using a tape and is at end-of-volume, it has a null priority.)
Status	The status of the request. A value of `active` indicates that the mover is processing the chunks assigned to it; any other value indicates why the request is not currently being processed. Only the `active` status is used for write movers because they are not created until both a volume and a drive are available at the same time.
Type	The type of data-mover process, either `read` or `write`.
Volume Group	The VG that contains the volume.
VSN	VSN of the volume

Fields for the MSP Request Database Table

Table 17-9 describes the fields available for the **MSP Request** (`msp_request`) database table.

Table 17-9 MSP Request Fields

Field	Description
Daemon Request ID	The daemon request ID that created this MSP request.
Library Server	The LS associated with this request. For a disk MSP or a DCM MSP, this will be null.
MSP Destination Offset	The byte location in the file where recalled data will be written. This field is only used for copy daemon requests and will be null in all other cases.
MSP Key	The string used to identify the MSP/VG copy to be used for the recall. For migrate requests, this field will be null.
MSP Length	Length of the data transfer. For migrate requests, this will always match the file's length.
MSP Request ID	A number that maps this record in the `msp_request` table to the `chunk` table.
MSP Request Type	The type of this request, either `migrate` or `recall`.
MSP Source Offset	The byte location in the file where the data transfer will begin. For migrate requests, this will always be `0`.
Priority	Priority of the request, in the range `1` (lowest priority) through `1024` (highest priority). The default priority is `512`.
Volume Group	The VG associated with this request.
Start Time	The time at which this request was sent to the MSP/VG.

SGI LTFS

This chapter discusses the SGI Linear Tape File System (LTFS):

- "SGI LTFS Features" on page 631
- "SGI LTFS Requirements" on page 632
- "Best Practices for SGI LTFS" on page 632
- "SGI LTFS Configuration" on page 633
- "`sgi-ltfs` Command Examples" on page 637

Also see "LTFS in DMF Manager" on page 263.

SGI LTFS Features

LTFS lets you access tape media with an interface that behaves like a hard disk. The content of an LTFS tape appears to users as a disk directory listing, which is easy to use with standard file operations. Because it is a tape format standard, LTFS lets you share data across platforms, similar to using a USB drive or memory stick.

SGI LTFS software provides LTFS capability for the following:

- Tapes: Ultrium5-1500, Ultrium6-2500, T10000T2
- Drives: LTO-5, LTO-6, T10000C, T10000D

SGI LTFS lets you do the following:

- Show information about LTFS cartridges / drives and configuration.
- Import new cartridges into the LTFS group.
- Format, mount, or unmount the LTFS on the imported cartridges.
- Eject a cartridge from a library managed by OpenVault.
- Reserve a drive exclusively for LTFS use.
- Borrow the drive from a drive group that belongs to another application, and then return the drive group after the operation completes. For example, you can

configure SGI LTFS so that it may temporarily borrow a drive from certain DMF drive groups. If an operation is issued when no drives in the reserved group are available, SGI LTFS will find a compatible drive in the list of specified DMF drive groups; after the operation completes, it will return the drive to DMF.

- Purge a cartridge record in the OpenVault catalog.
- Check an LTFS cartridge consistency.

SGI LTFS Requirements

SGI LTFS is installed by the `sgi-ltfs` package. It is implemented based on HP LTFS 2.0.0.0 and is GPL licensed. It requires the OpenVault mounting service and runs on the host that is the OpenVault server. The drives to be used by LTFS must be accessible from the OpenVault server.

Best Practices for SGI LTFS

This section discusses the following:

- "Avoid Parallel Use" on page 632
- "Verify Copies are Present Before Deleting Originals" on page 632
- "Ensure that Files are Online Before Copying to LTFS" on page 633

Avoid Parallel Use

A tape drive provides sequential-access storage. For best performance, only one user / process should access a given LTFS tape at any point in time; multiple users / processes accessing in parallel will result in poor performance.

Verify Copies are Present Before Deleting Originals

When you copy files to an LTFS filesystem, the data is held in filesystem cache until a sufficient amount is present; at that point, the data is copied to tape. If an error is present, it will not occur until the write to tape actually takes place. Therefore, you should always use the `ls`(1) command to verify that files are present in the LTFS filesystem before removing the original files.

Ensure that Files are Online Before Copying to LTFS

You should ensure that files are online (that is, regular or dual-state) before copying them to LTFS. For example, you could use the following `dmget`(1) command to ensure that all of the files in `/mydir` are online:

```
# find /mydir -type -f print | dmget -a
```

After this completes, you can use the `cp`(1) command or any drag-and-drop interface to copy the online files from the filesystem managed by DMF to the LTFS filesystem.

Note: Recalling offline files before using drag-and-drop will make the copy process more efficient.

SGI LTFS Configuration

This section discusses the following:

- "`sgi-ltfs.conf` File Format" on page 633
- "`sgi-ltfs.conf` Parameters" on page 634
- "`sgi-ltfs.conf` Examples" on page 635

`sgi-ltfs.conf` File Format

The `/etc/sgi-ltfs.conf` file contains configuration parameters and their values. You can modify the file directly or by using the DMF Manager interface (see "LTFS in DMF Manager" on page 263). This file will be preserved when SGI LTFS is updated to a newer version.

The file consists of ASCII case-sensitive strings separated by white space in the following format:

PARAMETER_NAME *value*

The # character indicates that the remainder of a line is a comment.

Note: Only comments in the leading comment block will be preserved when configuring with DMF Manager.

Most parameters have default values. The order in which parameters are listed is not important.

sgi-ltfs.conf Parameters

The /etc/sgi-ltfs.conf file can contain the following parameters:

BORROW_FROM_DG

Specifies a comma-separated list of one or more OpenVault drive groups that are owned by another application (such as DMF) from which LTFS may borrow drives. This parameter optionally also specifies the maximum number of drives that may be borrowed from the drive groups. The syntax is as follows (note that there is no space after the comma):

name1[,*name2*...] [*max_drives*]

You can include multiple instances of this parameter in order to specify different maximum numbers of drives for different sets of drive groups; the *max_drives* value applies to each drive group listed for a given BORROW_FROM_DG parameter. The default for *max_drives* is 1.

If you do not specify the parameter, LTFS will not borrow any drives and will use drives only from the drive group specified for LTFS_DRIVE_GROUP.

LTFS_CART_GROUP

Specifies the single OpenVault cartridge group to be reserved for use by LTFS. The default is ltfs_carts.

LTFS_DRIVE_GROUP

Specifies the single OpenVault drive group to be reserved for use by LTFS. The sgi-ltfs command will create this drive group for you if it does not already exist. The default is ltfs_dg.

Caution: After you have configured LTFS, you should be careful if you change this parameter. If you redefine this parameter but the old drive group is not empty, the drives in the old group will be unavailable to the sgi-ltfs command.

LTFS_MAX_DRIVES	Specifies the maximum number of drives that can be in the drive group specified by LTFS_DRIVE_GROUP at any given time. The default is 2.
LTFS_MOUNT_ROOT	Specifies the path that will be added to the tape name to form the mount point of the LTFS filesystem. The default is /dmf/ltfs. For example, an LTFS filesystem for tape vsn123 will be mounted as /dmf/ltfs/vsn123 by default.
LTFS_OV_KEYFILE	Specifies the path to the keyfile used by LTFS when connecting to the OpenVault server. The default is /dmf/home/ltfs_keys.
OV_SERVER	Specifies the name associated with the IP address on which the OpenVault server is listening The default is the hostname where you run the sgi-ltfs command.

sgi-ltfs.conf Examples

This section discusses the following:

- "Borrowing Drives (Simple)" on page 635
- "Borrowing Drives (More Complex)" on page 636
- "Using the LTFS Drive" on page 636

Borrowing Drives (Simple)

The following example uses default values for most of the parameters but permits up to two drives to be borrowed from the dg1 drive group, which is owned by another application:

```
BORROW_FROM_DG      dg1
```

Borrowing Drives (More Complex)

The following example borrows drives, using a more complex scenario:

```
OV_SERVER      my_host
LTFS_CART_GROUP       my_ltfs_cart_group
LTFS_DRIVE_GROUP      my_ltfs_dg
LTFS_MAX_DRIVES       3
LTFS_MOUNT_ROOT       /usr/ltfs
LTFS_OV_KEYFILE       /somewhere/ltfs_keys
BORROW_FROM_DG      dg1,dg2
BORROW_FROM_DG      dg3 3
BORROW_FROM_DG      dg4,dg5 2
```

The above example shows the following:

- The name of the OpenVault server is `my_host`
- The OpenVault cartridge group `my_ltfs_cart_group` and up to three drives in the OpenVault drive group `my_ltfs_dg` will be reserved for LTFS use
- The OpenVault keyfile is `/somewhere/ltfs_keys`
- The mount point of the LTFS filesystem will begin with `/usr/ltfs`
- In addition to the drives reserved above for LTFS use in `my_ltfs_dg`, LTFS can borrow drives that are normally managed by DMF as follows, up to a maximum of three borrowed drives total:
  - One drive (the default) from `dg1` or `dg2`
  - Three drives from `dg3`
  - Two drives from `dg4` or `dg5`

  For example, LTFS could borrow two drives from `dg4` and one drive from `dg5`

Using the LTFS Drive

The following example is similar, except that it will use the host where `sgi-ltfs` is executed as the OpenVault server and it will not borrow any drives (because there is

no BORROW_FROM_DG parameter). It will also use the default maximum of two drives from the my_ltfs_dg drive group:

```
LTFS_CART_GROUP       my_ltfs_cart_group
LTFS_DRIVE_GROUP      my_ltfs_dg
LTFS_MOUNT_ROOT       /usr/ltfs
LTFS_OV_KEYFILE       /somewhere/ltfs_keys
```

sgi-ltfs Command Examples

This section discusses the following:

- "Reading an LTFS Tape" on page 637
- "Reserving a Drive and Preparing a New LTFS Tape" on page 638
- "Reading a T10000C-Formatted Tape on a T10000D Drive" on page 639
- "Reformatting a T100002 Tape" on page 639

For details about the sgi-ltfs(8) command, see the man page.

Reading an LTFS Tape

To read an LTFS tape that contains data, do the following:

1. Import the tape with physical cartridge label PCL22 to the LTFS group, which also requires that you specify the cartridge-type name:

   For example, for a tape named of type T10000T2:

   ```
   # sgi-ltfs -i PCL22 -T T10000T2
   ```

2. Mount the tape as an LTFS filesystem:

   ```
   # sgi-ltfs -m PCL22
   ```

   If no suitable drive is found in the LTFS group, sgi-ltfs will search drive groups that belong to other applications if they are listed as values for the BORROW_FROM_DG parameter in the /etc/sgi-ltfs.conf file (see "sgi-ltfs.conf Parameters" on page 634). By default, the filesystem will be mounted as /dmf/ltfs/PCL22.

3. Copy files from the tape:

   # **cp /dmf/ltfs/PCL22/* /my_dir**

4. Unmount the filesystem:

   # **sgi-ltfs -u PCL22**

   If the drive was borrowed, it will be returned to the original drive group.

Reserving a Drive and Preparing a New LTFS Tape

To reserve `drive2` that belongs to DMF and prepare a new LTFS tape, do the following:

1. Ensure that `drive2` is in one of the drive groups listed for the `BORROW_FROM_DG` parameter in the `/etc/sgi-ltfs.conf` file (see "`sgi-ltfs.conf` Parameters" on page 634).

2. Reserve the drive:

   # **sgi-ltfs -r drive2**

3. Format the tape named `PCL33`:

   # **sgi-ltfs -f PCL33 -d drive2**

4. Mount the tape:

   # **sgi-ltfs -m PCL33 -d drive2**

5. Copy data to the tape:

   # **cp my_dir/* /dmf/ltfs/PCL33**

6. Make sure that the files are on the tape:

   # **ls /dmf/ltfs/PCL33**

7. Unmount the tape:

   # **sgi-ltfs -u PCL33**

8. Eject the tape from the library:

   # **sgi-ltfs -e PCL33**

9. Purge the tape information in the OpenVault system:

   # **sgi-ltfs -p PCL33**

   At this point, the tape can be reused.

Reading a T10000C-Formatted Tape on a T10000D Drive

A tape formatted on a T10000C drive can be mounted read-only (using the `-n` option) on a T10000D drive.

Note: A tape formatted on a T10000D drive cannot be mounted on T10000C drive.

For example, suppose that the tape named `PCL33` was formatted on a T10000C drive. To mount it on a T10000D drive, enter the following:

sgi-ltfs -m PCL33 -d t10kd -n

Reformatting a T100002 Tape

To reformat a T100002 tape, specify the compatible drive and the force option (`-o` or `--force`). For example, if you want to change `PCL33` from T10000C-format to T10000D-format:

sgi-ltfs -f PCL33 -d t10kd -o

Troubleshooting

This chapter contains the following:

- "Replacing a Bad DMF Copy with a Good DMF Copy" on page 642
- "Restoring Files from Backups" on page 643
- "Recovery and Restoration of DMF Database" on page 646
- "Backup Consistency Errors" on page 650
- "Disaster Recovery Procedures" on page 650
- "Filesystem Errors" on page 658
- "Unable to Use the `dmi` Mount Option" on page 660
- "EOT Error" on page 660
- "Tape Drive Not Claimed by `ts`" on page 660
- "Drive Entry Does Not Correspond to an Existing Drive (OpenVault)" on page 660
- "Drive Does Not Exist (TMF)" on page 661
- "No Usable Drives" on page 661
- "DMF Manager Errors" on page 666
- "Delay In Accessing Files in an SMB/CIFS Network Share" on page 668
- "Operations Timeout or Abort on Windows" on page 669
- "Windows Explorer Hangs" on page 669
- "Poor Migration Performance" on page 669
- "Poor Recall Performance" on page 669
- "Remote Connection Failures" on page 670
- "yast2 Disk Space Warning" on page 670
- "Linux CXFS Clients Cannot Mount Managed Filesystems" on page 670
- "Mount Errors for a Drive" on page 671

- "Verifying the Integrity of Library Server Volumes" on page 672
- "dmemptytape Process Did Not Complete" on page 674
- "DMF Database Server Startup Issues" on page 674
- "Using SGI Knowledgebase" on page 675
- "Reporting Problems to SGI" on page 675

Also see the troubleshooting chapter of the *DMF 6 Client Guide for Windows Systems.*

Replacing a Bad DMF Copy with a Good DMF Copy

If there are multiple copies of a file on secondary storage and one of them becomes corrupted, you can replace the corrupt copy with a good copy by using the dmmove(8) command. You can do this by supplying any of the following via standard input:

- The file handle
- BFID (by adding the -b option)
- Pathname (by adding the -p option)

For example, suppose that the copy of /dmfusr1/myfile has been corrupted on volume group vg1, but there is a good copy on vg2. You would do the following:

1. Delete (-d) the bad copy from vg1 and retain the good copy on vg2 (using -p to specify the file by its pathname supplied by standard input):

   ```
   # echo "/dmfusr1/myfile" | dmmove -d vg1 -p vg2
   ```

2. Delete no copies, place a good copy on vg1, and retain the good copy on vg2:

   ```
   # echo "/dmfusr1/myfile" | dmmove -d none -p vg1 vg2
   ```

   Note: Using best practices, you should always explicitly state all of the MSPs/VGs from which a copy should be deleted (none in this case) and all of the MSPs/VGs into which a copy should be placed/retained. In this case, you must include vg2 on the command line in order to retain the copy on vg2.

For more information, see the dmmove(8) and dmattr(8) man pages.

Restoring Files from Backups

This section discusses the following:

- "Overview of Restoring Files" on page 643
- "Ensuring Accuracy with `xfsdump`" on page 645
- "Backing Up and Restoring Files without the DMF Scripts" on page 645
- "Filesystem Consistency with `xfsrestore`" on page 646

Overview of Restoring Files

To restore files, use the `xfsrestore`(8) command or the **Recovery** tab in DMF Manager; see "Restoring Filesystems in DMF Manager" on page 255.

Note: The `xfsrestore` command may attempt to read, write, or delete files that are under DMF management. If this occurs while DMF is not running, the `xfsrestore` process may block indefinitely waiting for a DMF event to be completed. If you use `xfsrestore` to create or modify files in a filesystem that already contains files managed by DMF, you are more likely to encounter this issue than if you use `xfsrestore` to populate an empty filesystem. To avoid this problem, use `xfsrestore` while DMF is running.

The `xfsdump`(8) and `xfsrestore`(8) commands back up filesystems. These utilities are designed to perform the backup function quickly and with minimal system overhead. They operate with DMF in two ways:

- When `xfsdump` encounters an offline file, it does not cause the associated data to be recalled. This distinguishes the utility from `tar`(1) and `cpio`(1), both of which cause the file to be recalled when they reference an offline file.
- The `dmmigrate`(8) command lets you implement a 100% migration policy that does not interfere with customary management of space thresholds.

  The `xfsdump` command supports the `-a` option specifically for DMF. If you specify the `-a` option, `xfsdump` will back up DMF dual-state (`DUL`) files as if they were offline (`OFL`) files. That is, when `xfsdump` detects a file that is backed up by DMF, it retains only the inode for that file because DMF already has a copy of the data itself. This dramatically reduces the amount of space needed to back up a filesystem and it also reduces the time taken to complete the backup, thereby

minimizing the chances of it being inaccurate due to activity elsewhere in the system. An added advantage of using `-a` is that files that are actively being recalled will still be backed up correctly by `xfsdump` because it does not need to copy the file's data bytes to secondary storage.

You can also use `dmmigrate` to force data copies held only in a disk cache manager (DCM) MSP cache to be copied to tapes/MAID in the underlying volume groups (VGs). This removes the need to back up the cache filesystem.

Most installations periodically do a full (level 0) backup of filesystems. Incremental backups (levels 1 through 9) are done between full backups; these may happen once per day or several times per day. You can continue this practice after DMF is enabled. When a file is migrated (or recalled), the inode change time is updated. The inode change time ensures that the file gets backed up at the time of the next incremental backup.

To automatically manage backup media, DMF includes configurable administrative scripts called `run_full_dump.sh` and `run_partial_dump.sh`, which employ `xfsdump` to backup to tape or disk. The scripts perform the following actions:

- *(optional)* Migrates all eligible files to dual-state
- *(optional)* Copies all eligible DCM MSP files on a DCM MSP system to dual-residency state
- Performs a database snapshot using `dmsnap`
- Backs up the directory containing that snapshot
- Backs up other filesystems
- After a successful full backup, frees up old backup media and disk space for future reuse

DMF also supports a matching wrapper around `xfsrestore` named `dmxfsrestore` to be used when restoring files that were backed up by these scripts. See the `dmxfsrestore`(8) man page for more information on running the command.

You can configure tasks in the `dump_tasks` object to automatically do full and incremental backups of the managed filesystems. See "`taskgroup` Object" on page 304 for more information.

For more information about parameters, see "Starting and Stopping the DMF Environment" on page 181.

Sites using OpenVault can add new backup media by using `dmov_makecarts` and/or `dmov_loadtapes` by providing the name of the task group as a parameter. Sites using TMF do not need any special steps to add new tapes, as TMF does not record details of which tapes are available to it.

Recycling old backup media is performed automatically after the successful completion of a full backup. In certain situations, such as running out of backup media, this pruning must be done manually by running `dmxfsprune`.

Ensuring Accuracy with `xfsdump`

The `xfsdump` program is written such that it assumes backups will only be taken within filesystems that are not actively changing. `xfsdump` cannot detect that a file has changed while it is being backed up, so if a user should modify a file while it is being read by `xfsdump`, it is possible for the backup copy of the file to be inaccurate.

To ensure that all file backup copies are accurate, perform the following steps when using `xfsdump` to back up files within a DMF filesystem:

1. Make sure that there is no user activity within the filesystem.
2. Ensure that DMF is not actively migrating files within the filesystem.
3. Run `xfsdump`, preferably with the `-a` option.

Backing Up and Restoring Files without the DMF Scripts

If you choose to back up and restore DMF filesystems without using the provided DMF scripts, there are several items that you must remember:

- The DMF scripts use `xfsdump` with the `-a` option to back up only data not backed up by DMF. You may also wish to consider using the `-a` option on `xfsdump` when backing up DMF filesystems manually.
- **Do not use the `-A` option** on either `xfsdump` or `xfsrestore`. The `-A` option avoids backing up or restoring extended attribute information. DMF information is stored within files as extended attributes, so if you do use `-A`, migrated files restored from that backup media will not be recallable by DMF.
- When restoring migrated files using `xfsrestore`, you must specify the `-D` option in order to guarantee that all information related to DMF is correctly restored.

Filesystem Consistency with `xfsrestore`

When you restore files, you might be restoring some inodes containing BFIDs that were soft-deleted since the time the backup was taken. (For information about soft-deletes, see "Cleaning Up Obsolete Database Entries" on page 586.) `dmaudit`(8) will report this as an inconsistency between the filesystem and the daemon database, indicating that the database entry should not be soft-deleted.

Another form of inconsistency occurs if you happen to duplicate offline or dual-state files by restoring all or part of an existing directory into another directory. In this case, `dmaudit` will report as an inconsistency that two files share the same BFID. If one of the files is subsequently deleted causing the database entry to be soft-deleted, the `dmaudit`-reported inconsistency will change to the type described in the previous paragraph.

While these `dmaudit`-reported inconsistencies may seem serious, there is no risk of losing user data. The `dmhdelete`(8) program responsible for removing unused database entries always first scans all managed filesystems to make sure that there are no remaining files that reference the database entries it is about to remove. It is able to detect either of these inconsistencies and will not remove the database entries if inconsistencies are found.

Be aware that inconsistencies between a filesystem and the daemon database can occur as a result of restoring migrated files. It is good practice to run `dmaudit` after every restore to correct those inconsistencies.

Recovery and Restoration of DMF Database

The basic strategy for recovering a lost or damaged DMF database is to recreate it by applying journal records to a backup copy of the database. For this reason it is essential that database backup copies and journal files reside on a different physical device from the production databases; it is also highly desirable that these devices have different controllers and channels. The following sections discuss the database recovery strategy in more detail:

- "Database Backups" on page 647
- "Database Recovery Procedure" on page 647
- "Database Restore Procedure for Integrated Backups" on page 650

Database Backups

You can configure commands in the `run_copy_databases.sh` task (in the `dump_tasks` object) to automatically generate DMF database backups. See "`taskgroup` Object" on page 304, for more information.

You must back up the following files:

- The daemon database files and definition file in the *HOME_DIR/daemon_name* directory:

  ```
  dbrec.dat
  dbrec.keys
  pathseg.dat
  pathseg.keys
  dmd_db.dbd
  ```

Each LS database has the following files in the *HOME_DIR/ls_name* directory:

- CAT records:

  ```
  tpcrdm.dat
  tpcrdm.key1.keys
  tpcrdm.key2.keys
  ```

- VOL records:

  ```
  tpvrdm.dat
  tpvrdm.vsn.keys
  ```

- Database definition file: `libsrv_db.dbd`

Database Recovery Procedure

Note: You can also recover the database from a dump available in the **Available Dumps (Integrated Method)** pane in the **Recovery** tab of DMF Manager. See "Restore a Filesystem Component (Integrated Backups)" on page 255.

The DMF daemon and LS write journal file records for every database transaction. These files contain binary records that cannot be edited by normal methods and that must be applied to an existing database with the `dmdbrecover`(8) command. The following procedure explains how to recover the daemon database.

Warning: If you are running on multiple LSs, always ensure that you have the correct journals restored in the correct directories. Recovering a database with incorrect journals can cause irrecoverable problems.

Procedure 19-1 Recovering the Databases

If you lose a database through disk spindle failure or through some form of external corruption, use the following procedure to recover it:

1. Ensure that DMF is stopped. In a high-availability (HA) environment, see *High Availability Guide for SGI InfiniteStorage*. In a non-HA environment, execute the following:

   # **`service dmf stop`**

2. Do one of the following depending upon your circumstances:

   - If you have configured the `run_copy_databases` task, restore the files from the directory with the most recent copy of the databases that were in *HOME_DIR* to *HOME_DIR/daemon* or *HOME_DIR/LS_NAME*.
   - If you have **not** configured the `run_copy_databases` task, reload an old version of the daemon or LS database. Typically, these will be from the most recent dumps of your filesystem.

3. Ensure that the default *JOURNAL_DIR*/*daemon_name* (or *JOURNAL_DIR/ls_name*) directory contains all of the time-ordered journal files since the last update of the older database.

   For the daemon, the files are named `dmd_db.`*yyyymmdd[.hhmmss]*.

   For the LS, the journal files are named `libsrv_db.`*yyyymmdd[.hhmmss]*.

4. Use `dmdbrecover` to update the old database with the journal entries from journal files identified in step 3. The following commands result in the recovery of the daemon and library server databases, assuming the backup copy of the databases and the journals exist as specified by `/etc/dmf/dmf.conf`:

   # **`dmdbrecover -n`** ***`daemon_name`*** **`dmd_db`**
   # **`dmdbrecover -n`** ***`ls_name`*** **`libsrv_db`**

   Note: This process may take several hours to complete.

Example 19-1 Database Recovery

Suppose that the filesystem containing *HOME_DIR* was destroyed on February 1, 2004, and that your most recent backup copy of the daemon database and LS database is from January 28, 2004. To recover the databases, you would do the following:

1. Stop DMF (in an non-HA environment):

   # **service dmf stop**

2. Ensure that *JOURNAL_DIR/daemon_name* (or *JOURNAL_DIR/ls_name*) contains the following journal files (one or more for each day):

   JOURNAL_DIR/daemon_name

   ```
   dmd_db.20040128.235959
   dmd_db.20040129.235959
   dmd_db.20040130.235959
   dmd_db.20040131.235959
   dmd_db.20040201
   ```

   JOURNAL_DIR/ls_name

   ```
   libsrv_db.20040128.235959
   libsrv_db.20040129.235959
   libsrv_db.20040130.235959
   libsrv_db.20040131.235959
   libsrv_db.20040201
   ```

3. Restore databases from January 28, to *HOME_DIR*/*daemon_name* and/or *HOME_DIR/ls_name*. The following files should be present:

   HOME_DIR/*daemon_name*

   ```
   dbrec.dat
   dbrec.keys
   pathseg.dat
   pathseg.keys
   ```

   HOME_DIR/ls_name

   ```
   tpcrdm.dat
   tpcrdm.key1.keys
   tpcrdm.key2.keys
   tpvrdm.dat
   ```

```
tpcrdm.vsn.keys
```

4. Update the database files created in the step 3 by using the following commands, assuming that the name of the daemon database is `daemon` and the name of the library server is `ls`, as defined in `/etc/dmf/dmf.conf`:

```
# dmdbrecover -n daemon dmd_db
# dmdbrecover -n ls libsrv_db
```

Note: This process may take several hours to complete.

Database Restore Procedure for Integrated Backups

If you are using the integrated backup method, you can restore the DMF database from an online dump file. Select the most current dump file from the list under **Available Dumps (Integrated Method)** and select **Restore databases**. (You can also select an older database dump file and replay the journals.) For more information, see "Restore a Filesystem Component (Integrated Backups)" on page 255.

However, if no dump containing a good database is available, see "Disaster Recovery Procedures" on page 650.

Backup Consistency Errors

If a full or incremental backup produces consistency errors, you can run the `run_full_dump.sh` script manually with the appropriate arguments to prune orphaned filesystem dumps and orphaned inventory entries. See "`run_full_dump.sh`" on page 142.

Disaster Recovery Procedures

If the DMF database is lost or corrupt and the latest database backup file residing in the backup filesystem becomes inaccessible, you must first restore access to the backup filesystem and subsequently the DMF database to restore the server.

This section discusses the following:

- "Disaster Recovery for Basic DMF without HA" on page 651

- "Disaster Recovery for Parallel DMF" on page 655
- "Disaster Recovery for DMF with HA" on page 657

Disaster Recovery for Basic DMF without HA

Note: If a good copy of the DMF database is available from a dump file, see "Database Restore Procedure for Integrated Backups" on page 650.

If you are using the integrated backup method and basic DMF in an non-HA environment, you can use DMF Manager to perform disaster recovery.

You can use the following menu in DMF Manager:

Recovery
> Disaster Recovery

The above menu opens the **Disaster Recovery** dialog shown in Figure 19-1.

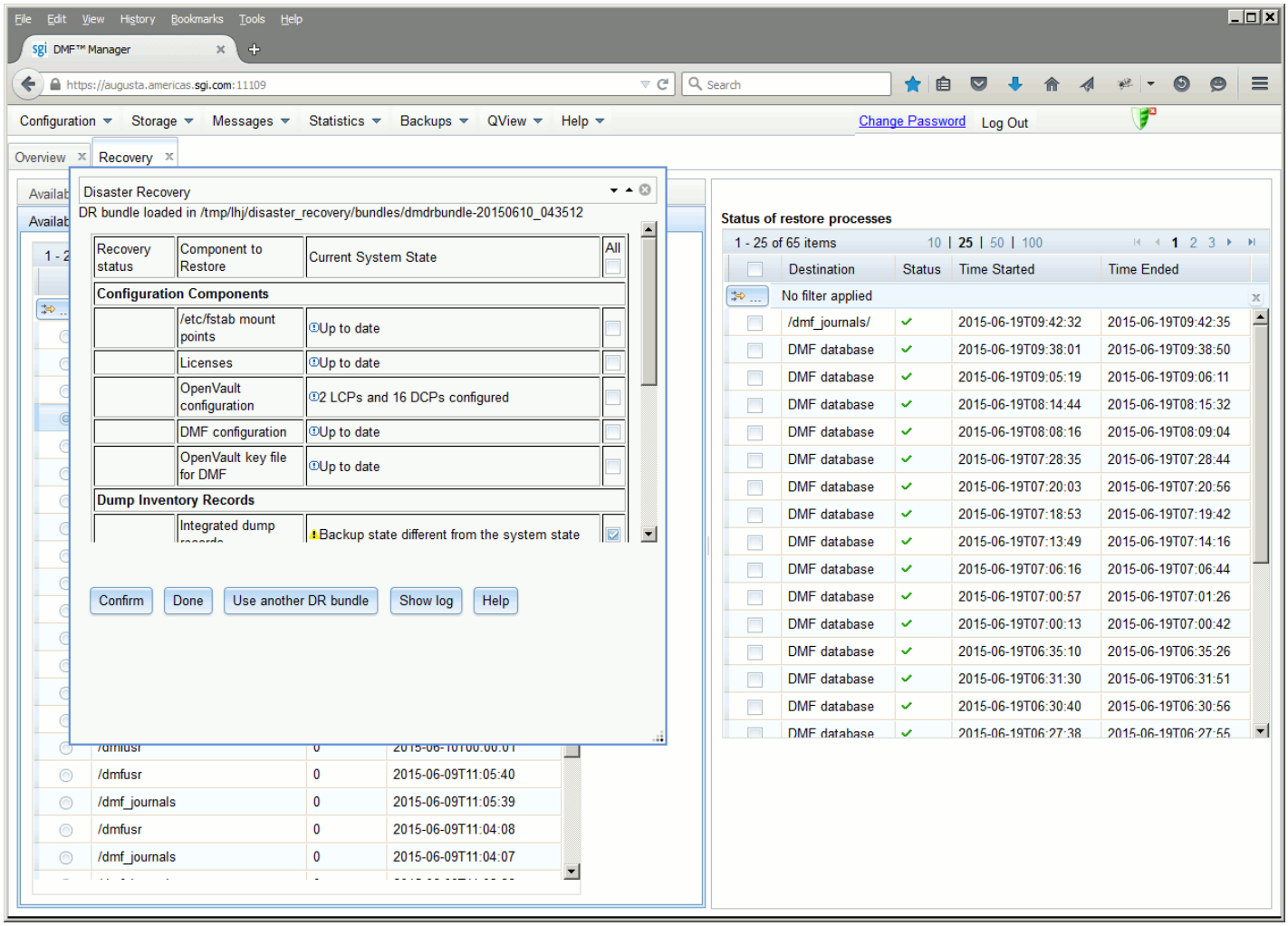

Figure 19-1 Disaster Recovery

Do the following:

1. Open the **Disaster Recovery** dialog and select **Confirm**.
2. Select the following:

   Backups
   > Recovery
   > Disaster Recovery...

3. Specify the required information:

   a. Specify the location of the DR bundle, such as `/hosts/nfsmirror/dmdrbundle-20150623_000000.tar.gz` or `root@remotehost:/drbundles/dmdrbundle-20150623_000000.tar.gz`.

   b. Specify the location to store temporary files during the recovery process The filesystem containing this directory should have enough space to hold the several components of the DR bundle, including the DR database (which is a collection of all metadata pertaining to the backup filesystem).

   c. Click **Load**.

4. Choose the components that you want to recover:

   - Configuration components:
     - `/etc/fstab` mount points
     - Licenses in the `/etc/lk/keys.dat` file
     - OpenVault configuration
     - DMF configuration
     - OpenVault key file for DMF
   - Dump inventory records:
     - Integrated dump records containing status of all completed dumps in a database residing in *HOME_DIR*`/backups/dumps.db`, which is required for access to and operations on dumps through DMF Manager
     - XFS inventory, which has information on all filesystem dumps in the `xfsdump` inventory residing in `/var/lib/xfsdump/inventory`.

Note: SGI recommends that you create `/var/lib/xfsdump` as a symbolic link to *HOME_DIR*`/site-specific/xfsdump`.

- Backup filesystem components:

Caution: Restoring these components could potentially remove the current DMF database. Use with extreme caution.

  - The backup filesystem defined by the `DUMP_DESTINATION` configuration parameter, such as `/dmf_dumps`
  - Disaster recovery database containing the metadata of all files residing in the `DUMP_DESTINATION` filesystem

- DMF database components

Caution: Restoring these components could potentially remove the current DMF database. Use with extreme caution. The disaster-recovery process tries to find any transactions in the journal filesystems that should be replayed; therefore, before trying this operation, you must make sure that the journal filesystem (specified by the `JOURNAL_DIR` parameter) is either intact or already restored.

Note: You should take care to restore all relevant components:

- If you restored only the DR database, you would be left with an incomplete version of DMF. If you restore the DR database, you must restore the DMF databases also.
- If you restore the DMF databases, you must also apply any available journals in order to restore all of the transactions that occurred after the database snapshot.

5. Click **Confirm** to accept the options and perform the disaster recovery.

Note: The server will store the state of the DR process. If your browser session is closed during the process, you can reopen DMF Manager later continue the rest of the process.

6. After the DR process completes, go to the **Overview** panel and check the status of DMF and OpenVault and restart them if necessary.

Disaster Recovery for Parallel DMF

If you are using the Parallel Data-Mover Option (Parallel DMF) in an non-HA environment, do the following for disaster recovery:

1. Install DMF software on the replacement DMF server (which is the same node as the OpenVault server, according to best practices for simplicity) and replacement parallel data-mover nodes, as directed in the *Release Note for ISSP*.

2. Run the `ov_makecert`(8) command on the DMF server:

   ```
   dmfserver# ov_makecert
   ```

3. Set up the systems to enable distributed recovery across the cluster:

   a. Copy the file `/opt/openvault/lib/cert.pem` from the DMF server to the passive DMF server and each parallel data-mover node.

   b. Configure passwordless `ssh` from the DMF server to each parallel data-mover node.

4. On the DMF server, start the `dmdr_admin`(8) command:

   ```
   dmfserver# dmdr_admin -l DR_bundle -d tmpdir
   ```

   where:

DR_bundle	Specifies the location of the disaster-recovery (DR) bundle, such as `/hosts/nfsmirror/dmdrbundle-20150623_000000.tar.gz` or `root@remotehost:/drbundles/dmdrbundle-20150623_000000.tar.gz`
tmpdir	Specifies a temporary location on the DMF server to use during the disaster-recovery process. The filesystem containing this directory should have enough space to hold the several components of the DR bundle, including the DR database (which is a

collection of all metadata pertaining to the backup filesystem).

For example:

```
dmfserver# dmdr_admin -l /hosts/nfsmirror/dmdrbundle-20150623_000000.tar.gz -d /tmp/dumptmp
```

5. At the `DR >` prompt in the interactive `dmdr_admin` session, do the following:

   a. Restore the network configuration (that is, the `/etc/sysconfig/network-scripts/ifcfg-*` files on RHEL or the `/etc/sysconfig/network/ifcfg-*` files on SLES) on all nodes:

      ```
      DR > restore config * network
      ```

   b. Restore the `/etc/hosts` file on all nodes:

      ```
      DR > restore config * host
      ```

   c. Restore the license file on the DMF server:

      ```
      DR > restore config servername licenses
      ```

      where

      servername Specifies the hostname of the DMF server

      For example, if the hostname is `dmfserver`:

      ```
      DR > restore config dmfserver licenses
      ```

   d. Restore CXFS:

      ```
      DR > restore cxfs
      ```

   e. Exit the tool:

      ```
      DR > quit
      ```

6. Reboot the DMF server.
7. Reboot each parallel data-mover node.
8. From the DMF server, restore OpenVault on each parallel data-mover node:

   a. Restart the `dmdr_admin` command on the DMF server. For example:

```
dmfserver# dmdr_admin -l /hosts/nfsmirror/dmdrbundle-20150623_000000.tar.gz -d /tmp/dumptmp
```

b. Restore the DMF configuration file on the DMF server and all parallel data-mover nodes:

```
DR > restore config * DMF
```

c. Restore the OpenVault configuration for each parallel data-mover node:

```
DR > restore config movername OpenVault
```

where

movername	Specifies the hostname of the parallel data-mover node

For example, for the node named `mover1`:

```
DR > restore config mover1 OpenVault
```

Repeat this command for each parallel data-mover node.

d. After the OpenVault configuration is restored for each parallel data-mover node, remove the temporary files from the *tmpdir* location:

```
DR > unload
```

e. Exit the tool:

```
DR > quit
```

9. Restore the following on the DMF server using the **Recovery** page of DMF Manager:

   - `xfsdump` inventory
   - Database for the dumps produced by the integrated backup processes (*HOME_DIR*`/backups/dumps.db`)

10. If required, use the **Recovery** page to restore other components, such as `/etc/fstab` mount entries and other filesystems from the dumps

For further assistance, call SGI Professional Services.

Disaster Recovery for DMF with HA

If you are using DMF with HA, call SGI Professional Services for assistance with disaster recovery.

Filesystem Errors

If the filesystems required for the DMF administrative directories are not mounted when you try to apply configuration changes using DMF Manager or when you use dmcheck, you will see errors such as the following:

```
  ERROR:  Directory for JOURNAL_DIR (/dmf_journals/journals) does not exist.
  ERROR:  MOVE_FS "/dmf/move"   must be a filesystem root
  ERROR:  Filesystem "/dmi_fs" is not mounted.
  ERROR:  A DCM's STORE_DIRECTORY (/dmf/dcm_name_store) must be a filesystem root.
  ERROR:  Filesystem "/" is not a DMAPI filesystem
  ERROR:  No such directory /dmf_journals/database_copies.
  ERROR:  OpenVault server is not up or client is misconfigured.
```

For example, following is the complete output from dmcheck:

```
# dmcheck

Checking DMF installation.
   Linux mynode 3.0.31-0.9-default #1 SMP Tue May 22 21:44:30 UTC 2012
(2dc3831) x86_64 x86_64 x86_64 GNU/Linux - mynode
   SuSE-release: SUSE Linux Enterprise Server 11 (x86_64)
   SuSE-release: VERSION = 11
   SuSE-release: PATCHLEVEL = 2
   sgi-issp-release:     SGI InfiniteStorage Software Platform, version
2.6, Build 706r75.sles11sp2-1207112009
   lsb-release:
LSB_VERSION="core-2.0-noarch:core-3.2-noarch:core-4.0-noarch:core-2.0-x86_64:core-3.2-x86_64:core-4.0-x8
   DMF version 5.6.0 rpm dmf-5.6.0-sgi260r41.sles11sp2 installed.

Checking DMF config file
 Scanning for non-comment lines outside define/enddef pairs
 Scanning for DMF parameters without values
 Checking all objects for invalid names
 Checking base
  ERROR:  Directory for JOURNAL_DIR (/dmf_journals/journals) does not exist.
 Checking daemon
  ERROR:  MOVE_FS "/dmf/move" must be a filesystem root
 Checking policy cache_policy
 Checking policy space_policy
 Checking policy chooser_policy
```

```
 Checking policy optional_chooser_policy
 Checking filesystem /dmi_fs
  ERROR:  Filesystem "/dmi_fs" is not mounted.
  WARNING:  Filesystem "/dmi_fs" inode size of 256 is inefficient for DMF.
 Checking filesystem /dmi_fs2
 Checking MSP msp
 Checking MSP cache (DCM-mode)
  ERROR:  A DCM's STORE_DIRECTORY (/dmf/dcm_name_store) must be a filesystem root.
  ERROR:  Filesystem "/" is not a DMAPI filesystem
 Checking MSP cache1 (DCM-mode)
 Checking Library Server ov_lib
 Checking Resource Watcher rw
 Checking Drive Group ov_drv
 Checking Volume Group volume1
  WARNING:  Please consider setting ZONE_SIZE to improve write performance.
  See the dmf.conf(5) man page for more information.
 Checking Volume Group volume2
 Checking Resource Scheduler ov_drvrs
 Checking Services dmf_services
 Checking Task Group vgtasks
 Checking Task Group daemon_tasks
  ERROR:  No such directory /dmf_journals/database_copies.
 Checking Task Group dump_tasks
 Checking Task Group library_tasks
 Checking Task Group node_tasks
 Checking for unreferenced objects
  WARNING:  Unreferenced watcher rw.
 Cross-checking LSs and task groups for duplicate VSNs

Checking other daemons.
 Checking OpenVault
  ERROR:  OpenVault server is not up or client is misconfigured.
 Checking chkconfig

7 errors found.
3 warnings found.
```

To resolve these problems, you must make and mount the filesystems required for the DMF administrative directories. See:

- "Configure Filesystems and Directories Appropriately for DMF" on page 94

- "Overview of the Installation and Configuration Steps" on page 165

Unable to Use the dmi Mount Option

By default, DMAPI is turned off on SLES 10 systems. If you try to mount with the `dmi` mount option, you will see errors such as the following:

```
kernel: XFS: unknown mount option [dmi]
```

See "Linux CXFS Clients Cannot Mount Managed Filesystems" on page 670.

EOT Error

A message of the following type means that there was no logical end-of-tape (EOT) mark written to the volume:

```
05:47:26-E   382537-dmatwc end_tape: NOTE: An EOT was not written to VSN 057751 prior to close
```

When DMF appends data to a volume, it positions to the EOT chunk in the EOT zone. Without a valid EOT chunk in the EOT zone, DMF might not be able to append to the volume; this may eventually cause the `HVFY` flag to be set. Set the `hsparse` flag on the volume to merge all the data off of it.

Tape Drive Not Claimed by ts

If a tape drive is not claimed by `ts`, see the `/var/log/messages` file for an indication as to why `ts` did not attach to a device.

Drive Entry Does Not Correspond to an Existing Drive (OpenVault)

If OpenVault starts before an HBA has discovered the devices, the devices will be unusable by OpenVault. In this case, you would see a message similar to the following:

```
Drive lto1_3 DCP lto1_3@boom config file scsi: entry does not correspond to an existing drive
```

You must add the HBA driver to the `/etc/sysconfig/kernel` file and restart OpenVault. See "Add HBA Drivers to the `initrd` Image" on page 107.

Drive Does Not Exist (TMF)

If a drive is not visible to TMF, it may be because an HBA device was not properly discovered. In this case, there would be a message in `/var/spool/tmf/daemon.stderr` such as the following:

```
File /dev/ts/pci0002:00:01.1/fc/500104f000700269-500104f00070026a/lun0 does not exist
```

You must add the HBA driver to the `/etc/sysconfig/kernel` file and restart TMF. See "Add HBA Drivers to the `initrd` Image" on page 107.

No Usable Drives

A variety of conditions can cause the drive group to report that no drives are available, such as shown in the following message:

```
DMF drive group dgc07 has had no usable drives in OpenVault drive group 'drives' for more than 0 seconds
```

To determine the source of the problem, check the following:

1. Is the OpenVault server is running? You can determine this by using the `ov_stat`(8) command. For example:

   ```
   # ov_stat

    The OV server 'node3' is UP.
   ```

   If it is not running, use the `ov_start`(8) command.

2. Is the OpenVault library that contains the drives usable? Use the `ov_stat -L` option. For example, the following shows that OpenVault library `t200` is usable:

   ```
   # ov_stat -l
   LibraryName Usable Disabled
   t120        false  temporary
   t200        true   false
   ```

3. Does the usable library have at least one usable drive? To determine this, do the following:

a. Use the ov_stat(8) command to see which drives are unusable in group drives specified in the error message. For example, to see the status of all drives:

```
# ov_stat -d
DriveName Group  Usable Access Disabled  SoftState HardState Occupied PCL
t200d1    drives false  true   temporary ready     unloaded false
t200d2    drives true   true   false     ready     unloaded false
t200d3    drives false  true   false     ready     unloaded false
```

The above output indicates that two drives (t200d1 and t200d3) are unusable. At least one reason for t200d1 not working is that it is temporarily disabled; you would use the following command to reenable it:

```
# ov_drive -E t200d1
```

You could then repeat the ov_stat -d command to see if t200d1 is usable.

There is no indication why t200d3 is unusable, so you must continue with the following step..

b. Determine why an enabled drive (in this case t200d3) is unusable by examining its DCPs with the ov_stat command. For example, to display the DCP status (-x) for all drives:

```
# ov_stat -dx
DCPName        DriveName Usable SoftState    Disabled  MsgLevel
t200d1@node1   t200d1    true   disconnected false     debug
t200d1@node2   t200d1    false  inactive     false     debug
t200d1_2@node1 t200d1    true   ready        false     debug
t200d1_2@node2 t200d1    false  inactive     false     debug
t200d2@node1   t200d2    false  disconnected temporary debug
t200d2@node2   t200d2    false  inactive     false     debug
t200d2_2@node1 t200d2    true   ready        false     debug
t200d2_2@node2 t200d2    false  inactive     false     debug
t200d3@node1   t200d3    false  disconnected permanent debug
t200d3@node2   t200d3    false  inactive     false     debug
t200d3_2@node1 t200d3    false  ready        temporary debug
t200d3_2@node2 t200d3    false  inactive     false     debug
```

The above output shows that none of the DCPs on node2 have been started, so they are not usable (because the SoftState value is inactive). All four DCPs for drive t200d3 are unusable, therefore drive t200d3 itself is unusable.

To fix the DCPs on node2, you would log on to host node2 and enter the following command to start those DCPs in an non-HA configuration:

```
# ov_start t200d1 t200d2 t200d3
```

Caution: If you are running in an HA configuration, do not start any DCPs or LCPs on the passive DMF server. In an HA environment, it is required that all DCPs and LCPs on the passive DMF server machine be in inactive state.

You could then repeat the above ov_stat command to verify the status of all DCPs:

```
# ov_stat -dx
DCPName         DriveName Usable SoftState    Disabled  MsgLevel
t200d1@node1    t200d1    true   disconnected false     debug
t200d1@node2    t200d1    true   disconnected false     debug
t200d1_2@node1 t200d1    true   ready        false     debug
t200d1_2@node2 t200d1    true   disconnected false     debug
t200d2@node1    t200d2    true   disconnected false     debug
t200d2@node2    t200d2    true   disconnected false     debug
t200d2_2@node1 t200d2    true   ready        false     debug
t200d2_2@node2 t200d2    true   disconnected false     debug
t200d3@node1    t200d3    false  disconnected temporary debug
t200d3@node2    t200d3    true   disconnected false     debug
t200d3_2@node1 t200d3    false  ready        temporary debug
t200d3_2@node2 t200d3    true   disconnected false     debug
```

According to the above output, all of the DCPs on node2 are now usable. However, there are still two DCPs on node1 that are unusable because they are disabled. The following ov_dcp(8) command enables all DCPs for all drives starting with the string t200:

```
# ov_dcp -E 't200.*' '.*'
```

Note: Although this may enable DCPs that are already enabled, that does not cause a problem.

You can then verify the status of all DCPs to determine if all DCPs on both nodes are usable:

```
# ov_stat -dx
DCPName         DriveName Usable SoftState    Disabled MsgLevel
t120d1@node1    t120d1    true   ready        false    debug
t120d1@node2    t120d1    true   disconnected false    debug
t120d2@node1    t120d2    true   ready        false    debug
t120d2@node2    t120d2    true   disconnected false    debug
t120d3@node1    t120d3    true   ready        false    debug
t120d3@node2    t120d3    true   disconnected false    debug
t120d4@node1    t120d4    true   ready        false    debug
t120d4@node2    t120d4    true   disconnected false    debug
t120d5@node1    t120d5    true   disconnected false    debug
t120d5@node2    t120d5    true   disconnected false    debug
t120d5_2@node1 t120d5     true   ready        false    debug
t120d5_2@node2 t120d5     true   disconnected false    debug
t120d6@node1    t120d6    true   disconnected false    debug
t120d6@node2    t120d6    true   disconnected false    debug
t120d6_2@node1 t120d6     true   ready        false    debug
t120d6_2@node2 t120d6     true   disconnected false    debug
t120d6_3@node1 T120d6     true   disconnected false    debug
t120d6_3@node2 t120d6     true   disconnected false    debug
t120d6_4@node1 t120d6     true   disconnected false    debug
t120d6_4@node2 t120d6     true   disconnected false    debug
t200d1@node1    t200d1    true   disconnected false    debug
t200d1@node2    t200d1    true   disconnected false    debug
t200d1_2@node1 t200d1     true   ready        false    debug
t200d1_2@node2 t200d1     true   disconnected false    debug
t200d2@node1    t200d2    true   disconnected false    debug
t200d2@node2    t200d2    true   disconnected false    debug
t200d2_2@node1 t200d2     true   ready        false    debug
t200d2_2@node2 t200d2     true   disconnected false    debug
t200d3@node1    t200d3    true   disconnected false    debug
t200d3@node2    t200d3    true   disconnected false    debug
t200d3_2@node1 t200d3     true   ready        false    debug
t200d3_2@node2 t200d3     true   disconnected false    debug
```

Conversely, you could use the `-y` option to limit the output to only the unusable drives and DCPs:

```
# ov_stat -dy
DriveName Group  Usable Access Disabled SoftState HardState Occupied PCL
#
# ov_stat -dxy
DCPName DriveName Usable SoftState Disabled MsgLevel
#
```

The lack out output for the above commands in this case confirms that there are no unusable drives or DCPs.

4. Are the LCPs for the library usable? For example:

```
# ov_stat -L t200 -x
LCPName      LibraryName Usable SoftState    Disabled  MsgLevel
t200@node3   t200        true   disconnected false     debug
t200_2@node3 t200        false  ready        permanent debug
```

The above output indicates that `t200@node3` is usable but `t200_2@node3` has been disabled. To reenable `t200_2@node3`, enter the following:

```
# ov_lcp -E t200 t200_2@node3
```

To verify that all libraries and LCPs are usable, add the `-y` option to restrict the output to unusable components. For example:

```
# ov_stat -L t200 -y
LibraryName Usable Disabled


# ov_stat -L t200 -xy
LCPName LibraryName Usable SoftState Disabled MsgLevel
```

The lack of output for the above commands in this case confirms that there are no unusable libraries or LCPs.

5. Can DMF start movers on the machines with working drives? Use the dmnode_admin(8) command to verify that the nodes are active. For example,

```
# dmnode_admin -l
Node Name  State        Enabled  Active Since          Dropouts
node1      Inactive     Yes      -                     0
node2      Disabled     No       -                     0
node3      Active       Yes      2014-Sep-03,15:23:53  0
```

The above output shows that node3 is the only node that is both active and enabled. However, the previous steps show that all of the DCPs are defined for either node1 or node2, but there are no drive connections to node3. Therefore no drives can be used until at least one of node1 or node2 is in Active state. See "Reenabling Parallel Data-Mover Nodes" on page 489.

For more information, see the dmnode_admin(8) man page and "Determining the State of Parallel Data-Mover Nodes" on page 488.

6. Can DMF communicate with OpenVault? To determine this, examine the following:
   - Output from the dmcheck(8) command
   - DMF library server log
   - OpenVault OVSERVER log
   - DMF Manager, as discused in "Displaying Node Status" on page 252

DMF Manager Errors

This section describes problems you may encounter when monitoring DMF with DMF Manager:

- "DMF Statistics are Unavailable Error Message" on page 666
- "DMF Statistics Graphs are Empty" on page 668
- "OpenVault Library Is Missing" on page 668

Also see "Filesystem Errors" on page 658.

DMF Statistics are Unavailable Error Message

```
This screen requires statistics from DMF that are unavailable;
check that DMF is running, including the "pmdadmf2"process.
Make sure the DMF "EXPORT_METRICS" configuration parameter is enabled.
```

If DMF statistics are unavailable, do the following:

1. Verify that the EXPORT_METRICS parameter in the base object of the /etc/dmf/dmf.conf file is set to ON.

If it is not, do the following:

Caution: Do not modify EXPORT_METRICS while DMF is running.

For instructions about starting and stopping DMF and the mounting service in an HA environment, see *High Availability Guide for SGI InfiniteStorage.*

a. Run the following commands as root to stop DMF:

```
# service dmf stop
```

b. Set EXPORT_METRICS to ON by editing the file or using DMF Manager.

c. Validate the change by using the dmcheck(8) command or DMF Manager

d. Start DMF:

```
# service dmf start
```

For more information, see:

- "Modifying an Object" on page 217
- "Validating Your Changes" on page 218
- "Displaying DMF Configuration File Parameters" on page 219
- "base Object" on page 277

2. Check that the data is passing through PCP by running the following command:

```
# pminfo -f dmf2.config.dmversion
```

For example, the following indicates that all is well:

```
# pminfo -f dmf2.config.dmversion

dmf2.config.dmversion
    value "6.0.0"
```

If no value is available, run the following commands as root to remove and reinstall the PCP performance metrics domain agents and restart DMF Manager:

```
# cd /var/lib/pcp/pmdas/dmf2
# ./Remove
# ./Install
```

```
# service dmfman restart
```

DMF Statistics Graphs are Empty

If the DMF Manager graphs are empty but you do not see the error message in "DMF Statistics are Unavailable Error Message" on page 666, verify that the `gmgrd` process is running:

```
dmfserver# ps -A | grep gmgrd
```

If it is not, restart the `gmgrd` process by restarting the `pcp-storage` service:

```
dmfserver# service pcp-storage restart
```

OpenVault Library Is Missing

```
No OpenVault-controlled library found.
```

This indicates that OpenVault is not running. Run the following command to verify that the `ov_stat` command is available:

```
# ls -lL /usr/bin/ov_stat
-rws--x--x 1 root sys 322304 Jul 22 2005 /usr/bin/ov_stat
```

If the file permissions are not `-rws--x--x` as shown above, run the following command to change the permissions:

```
# chmod 4711 /usr/bin/ov_stat
```

Delay In Accessing Files in an SMB/CIFS Network Share

If there is a delay in accessing files in an SMB/CIFS network share, it may be because the files are in a fully or partially offline state. The Windows Explorer desktop can be enabled to display a small black clock on top of a migrated file's normal icon; the black clock symbol indicates that there may be a delay in accessing the contents of the file. (This feature is disabled by default.) For more information, see "Modify Settings If Providing File Access via Samba" on page 127.

Operations Timeout or Abort on Windows

Operations such as `cp` can timeout on Windows systems or abort with the following message:

```
couldn't locate the origin file
```

This may occur if the `SessTimeout`, `ExtendedSessTimeout`, or `ServersWithExtendedSessTimeout` parameter is set to a value that is inappropriate for a DMF environment. See "Modify Settings If Providing File Access via Samba" on page 127.

Windows Explorer Hangs

If Windows Explorer hangs and the `no response ...` message appears in the Windows main title, it may be because the `SessTimeout`, `ExtendedSessTimeout`, or `ServersWithExtendedSessTimeout` parameter is set to a value that is inappropriate for a DMF environment. See "Modify Settings If Providing File Access via Samba" on page 127.

Poor Migration Performance

If you encounter poor migration performance, you can try to tune DMF's direct I/O size by modifying the `DIRECT_IO_SIZE` parameter for the `filesystem` object in the DMF configuration file (`/etc/dmf/dmf.conf`).

You can also try switching to buffered I/O migration by setting the `MIN_DIRECT_SIZE` parameter to a very large value.

See "`filesystem` Object" on page 350.

Poor Recall Performance

If you notice performance problems when recalling files, you can consider adjusting the `POSIX_FADVISE_SIZE` parameter in the `filesystem` object. This may make an improvement, but will require testing multiple factors such as the following:

- Memory pressure, which impacts buffer cache and forces the kernel to flush pages
- User filesystem activity

- The setting of various `sysctl vm` parameters (see the `Documentation/sysct/vm.txt` file in the Linux kernel source)

For more details about setting the `POSIX_FADVISE_SIZE` parameter in the DMF configuration file, see "`filesystem` Object Parameters" on page 351.

Note: Setting this parameter to a small, nonzero value may have an adverse affect on performance. See the `posix_fadvise`(2) man page for more information.

Remote Connection Failures

If there are an insufficient number of `xinetd tcpmux` instances configured, you may see remote connection failures. If this condition occurs, you will see messages like the following in the `/var/log/xinetd.log` file:

```
10/3/2@13:41:09: FAIL: tcpmux service_limit from=128.162.246.75
```

To solve this problem, see "Set the `xinetd tcpmux instances` Parameter Appropriately" on page 108.

yast2 Disk Space Warning

If you try to use `yast2` while RAID sets in the COPAN MAID are mounted by OpenVault, `yast2` displays a graphic that contains the following warning:

```
Warning: Disk space is running out!
```

However, the red disk usage shown for filesystems such as `/var/opt/openvault/clients/mounts/copan_C00d02` is expected; it indicates that the RAID sets are mounted by OpenVault. You can safely click the **OK** button at the bottom of the window.

Linux CXFS Clients Cannot Mount Managed Filesystems

In order for CXFS client-only nodes that are not running DMF software to mount CXFS filesystems, you must make manual modifications in order to run DMAPI. See the information about enabling DMAPI for Linux client-only nodes in the *CXFS 7 Client-Only Guide for SGI InfiniteStorage*.

Mount Errors for a Drive

If a drive is unable to mount a cartridge, it may be that the drive has firmware that is incompatible with the persistent reservation feature, which is on by default. In this case, you can disable persistent reservation on that drive by doing the following:

1. List the DCP paths for the problem drive. For example (output truncated):

```
# ov_dumptable -c DriveName,DCPHost,DCPControlPath DCP
C07d1     nodeA /dev/ts/pci0000:05:00.0/fc/2000001b3202a4dd-2100001b3202a4dd/lun1
C07d1     nodeB   /dev/ts/pci0000:07:00.1/fc/2001001b3222a4dd-2101001b3222a4dd/lun1
...
```

2. On each host, execute the following command to turn off persistent reservation on the path for that host to the problem drive, where *PATH* is the `DCPControlPath` as listed above:

   # **tsset -p off -d *PATH***

   For example, using the output above for `C07d1`, you would execute the following:

   - On host `nodeA`:

```
nodeA# tsset -p off -d /dev/ts/pci0000:05:00.0/fc/2000001b3202a4dd-2100001b3202a4dd/lun1
```

   - On host `nodeB`:

```
nodeB# tsset -p off -d /dev/ts/pci0000:07:00.1/fc/2001001b3222a4dd-2101001b3222a4dd/lun1
```

Note: Every path to a given drive used by OpenVault must be set the same way on every host (either all enabled or all disabled).

The drive will no longer experience the mount errors, but it will no longer have the data protection provided by the feature.

If in the future you install new firmware that supports persistent reservation, you can reenable the feature on the drive by doing the following on each host:

- Verify that the path has the persistent reservation feature disabled:

  # **tsmt -f *PATH* prsv**

If the feature is disabled, the above command will return a `Not Supported` error.

- Reenable the feature by entering the following on each host:

  # **tsset -p on -d *PATH***

Verifying the Integrity of Library Server Volumes

You can use the `dmatsnf`(8) command to verify the integrity of library server volumes. The command provides a number of options; see the man page for details.

The following example illustrates just a few `dmatsnf` options, in a series of steps in conjunction with the `dmcatadm`(8) command, that you can use to verify a volume (in this case, `C07IAI`) where some reads fail due to I/O errors. (To simplify the example, the volume was written with unrealistically small zones and no data was hard-deleted from this volume.)

1. List the chunks that are recorded in the database as being on the volume, using the `dmcatadm`(8) command:

```
# dmcatadm -m ls_maid -c 'list vsn=c07iai format zn,cn,vsn'
                          ZONE CHUNK
                     KEY   NUM   NUM VSN
-----------------------------------------
4d9dd5130000000000c6776b     1     1 C07IAI
4d9dd5130000000000c6776c     1     2 C07IAI
4d9dd5130000000000c6776d     1     3 C07IAI
4d9dd5130000000000c6776e     1     4 C07IAI
4d9dd5130000000000c6776f     2     6 C07IAI
4d9dd5130000000000c67770     2     7 C07IAI
```

2. Verify the volume:

```
# dmatsnf -v C07IAI -V vgc07
Processing vsn C07IAI
BOC: zn=1, cn=1, bfid=4d9dd5130000000000c6776b, file_offset=0, file_length=5000, expected_length=5000
BOC: zn=1, cn=2, bfid=4d9dd5130000000000c6776c, file_offset=0, file_length=5000000, expected_length=5000000
ERROR (C07IAI) at zn=1, cn=2, so=3139900: found block error, length=0: Input/output error

Skipped 51 bad blocks
ERROR (C07IAI): Too many consecutive block errors; will not try to read remainder of this zone. Consider increasing -
ERROR (C07IAI) at zn=1, cn=2, Database entry zn=1 cn=2 not read from tape due to preceding errors
ERROR (C07IAI) at zn=1, cn=2, Database entry zn=1 cn=3 not read from tape due to preceding errors
```

```
ERROR (C07IAI) at zn=1, cn=2, Database entry zn=1 cn=4 not read from tape due to preceding errors
BOC: zn=2, cn=6, bfid=4d9dd5130000000000c6776f, file_offset=0, file_length=1000000, expected_length=1000000
BOC: zn=2, cn=7, bfid=4d9dd5130000000000c67770, file_offset=0, file_length=27000000, expected_length=27000000
EOZ: zn=2, cn=8
--- TAPE MARK ---
EOI: zn=3, cn=9
7 errors found.
```

The above output shows that in the middle of chunk 2 (`cn=2`), `dmatsnf` saw 51 consecutive I/O errors. Because that exceeds the default maximum (for the `-e` option), `dmatsnf` quit reading data in zone `1` and positioned to zone 2 (`zn=2`), where it was able to successfully read the remainder of the volume.

To read all of zone 1, you must specify a maximum error count with `-e` that is equal to or greater than the number of blocks in the zone that have I/O errors; the easiest way to guarantee this is to use a value at least as large as the number of blocks in the volume, accomplished in the following steps.

3. Find the end-of-tape block ID value:

```
# dmvoladm -m ls_maid -c 'list vsn=c07iai format eb,ec,ez'
            EOT    EOT   EOT
VSN     BLOCKID  CHUNK  ZONE
----------------------------
C07IAI       d7      9     3
```

Note: The output is shown in hexadecimal.

4. Verify the volume again, but this time specify a maximum number of consecutive errors that is equal to the size of the volume by using the `-e` option with the decimal value of the end-of-tape block ID shown above (`d7` has a decimal value of `215`):

```
# dmatsnf -v C07IAI -V vgc07 -e 215
Processing vsn C07IAI
BOC: zn=1, cn=1, bfid=4d9dd5130000000000c6776b, file_offset=0, file_length=5000, expected_length=5000
BOC: zn=1, cn=2, bfid=4d9dd5130000000000c6776c, file_offset=0, file_length=5000000, expected_length=5000000
ERROR (C07IAI) at zn=1, cn=2, so=3139900: found block error, length=0: Input/output error

skipped 51 bad blocks
ERROR (C07IAI) at zn=1, cn=2, so=3139900: found cn=3, fo=24875824, DATA out of sequence
ERROR (C07IAI) at zn=1, cn=3, Database entry zn=1 cn=2 not found on tape
```

```
BOC: zn=1, cn=4, bfid=4d9dd5130000000000c6776e, file_offset=0, file_length=17000000, expected_length=17000000
EOZ: zn=1, cn=5
--- TAPE MARK ---
BOC: zn=2, cn=6, bfid=4d9dd5130000000000c6776f, file_offset=0, file_length=1000000, expected_length=1000000
BOC: zn=2, cn=7, bfid=4d9dd5130000000000c67770, file_offset=0, file_length=27000000, expected_length=27000000
EOZ: zn=2, cn=8
--- TAPE MARK ---
EOI: zn=3, cn=9
4 errors found.
```

The above output shows that in the middle of chunk 2, I/O errors were encountered. The following line means that `dmatsnf` was able to successfully read part of chunk 3 (the first part of chunk 3 had I/O errors):

```
ERROR (C07IAI) at zn=1, cn=2, so=3139900: found cn=3, fo=24875824, DATA out of sequence
```

`dmemptytape` Process Did Not Complete

An email message that contains wording similar to the following indicates that the `dmemptytape`(8) process was interrupted:

```
A dmemptytape process for vsn VSN did not complete.
```

The `dmemptytape` command uses a checkpoint feature so that you can restart it if it is interrupted, minimizing the chance that file copies will be lost. You should therefore reexecute the same `dmemptytape` command line in order to maintain all copies of all files. For details, see the `dmemptytape`(8) man page.

Caution: Do not manually remove any temporary files associated with `dmemptytape`. `dmemptytape` will automatically clean up the temporary directories after they are no longer needed.

DMF Database Server Startup Issues

If there are problems starting the DMF database server, do the following:

- Ensure that the `ident` service is started, according to the instructions in "Overview of the Installation and Configuration Steps" on page 165
- Restart the `xinetd` service if you have changed arguments

- Examine the `/var/log/dmf/svclog` file for issues related to starting up the DMF database server

Using SGI Knowledgebase

If you encounter problems and have an SGI support contract, you can log on to the SGI support website and access the **Search Knowledgebase** tool to help find answers.

See:

https://support.sgi.com

If you need further assistance, contact SGI Support.

Reporting Problems to SGI

As soon as you suspect a problem with DMF, run the following commands as `root` to gather relevant information about your DMF environment that will help you and SGI analyze the problem:

- Run the following command on the DMF server and every parallel data-mover node in order to gather system configuration information:

  # **`/usr/sbin/system_info_gather -A -o`** ***`nodename`*****`.out`**

- Run the following command once on the DMF server to collect information for today and the specified number of additional days (*previous-days* must be a numerical value greater than or equal to `0`):

  # **`dmcollect`** ***`previous-days`***

  Note: Take care to enter the correct number of previous days from which to gather information, so that logs containing the first signs of trouble are included in the collection.

  See the `dmcollect`(8) man page for additional information.

When you contact SGI Support, you will be provided with information on how and where to upload the collected information files for SGI analysis.

Messages

This appendix discusses the following:

- "`dmcatadm` Message Interpretation" on page 677
- "`dmvoladm` Message Interpretation" on page 679

If you are uncertain about how to correct these errors, contact your customer service representative.

`dmcatadm` Message Interpretation

The following lists the meaning of messages associated with the CAT records in the LS database:

nnn `bytes duplicated in volume group name`	Two or more chunks in the database, which belong to volume group (VG) name, contain data from the same region of the file.
`for vsn DMF001 chunk 77 chunkoffset < 0`	The `chunkoffset` value for chunk 77 on volume serial number (VSN) `DMF001` is obviously bad because it is less than 0.
`for vsn DMF001 chunk 77 chunklength < 0`	The `chunklength` value for chunk 77 on VSN `DMF001` is obviously bad because it is less than 0.
`for vsn DMF001 chunk 77 chunknumber < 0`	The `chunknumber` value for chunk 77 on VSN `DMF001` is obviously bad because it is less than 0.
`for vsn DMF001 chunk 77 filesize < 0`	The `filesize` value for chunk 77 on DMF001 is obviously bad because it is less than 0.

`for vsn DMF001 chunk 77 filesize < chunklength + chunkoffset`	The value of `chunklength` plus `chunkoffset` should be less than or equal to the `filesize`. Therefore, one or more of these values is wrong.
`for vsn DMF001 chunk 77 missing or improper vsn`	The list of VSNs for the chunk is improperly constructed. The list should contain one or more six-character names separated by colons.
`for vsn DMF001 chunk 77 zonenumber < 0`	The `zonenumber` value for chunk `77` on `DMF001` is obviously bad because it is less than `0`.
`for vsn DMF001 chunk 77 zonenumber > chunknumber`	Either the `zonenumber` value or the `chunknumber` value for chunk `77` on `DMF001` is wrong, because the `zonenumber` is larger than the `chunknumber` value. (Each zone contains at least two chunks, because the end-of-zone header on the volume counts as a chunk.)
`for vsn DMF001 chunk 77 filesize != file size in daemon entry (nnn)`	The `filesize` value in the chunk entry is different from the file size in the daemon record. If no daemon record was provided, this message indicates that more than one chunk exists for the BFID and that the `filesize` value is not the same for all the chunks.
`missing from cat db`	No corresponding CAT record was found for an existing daemon record.
`entry for volume group name missing from daemon db`	No corresponding daemon record was found for an existing CAT record.
`for volgrp name; no chunk for bytes nnn - nnn`	There is no chunk that contains the specified bytes of the file.

`dmvoladm` Message Interpretation

The following lists the meaning of messages associated with the VOL records in the LS database.

`blocksize is bad`	The `blocksize` field for the volume is less than or equal to 0.
`eotpos < largest position in cat (3746)`	The position for the end-of-volume (historically known as *EOT* for *end-of-tape*) descriptor on the volume is less than the largest position of all the chunk entries for the volume.
`chunksleft != number of cat chunks (256)`	The number of chunks referencing the volume in the CAT table does not equal the number of chunks left recorded in the VOL entry for the volume.
`dataleft != sum of cat chunk lengths (4.562104mb)`	The sum of the chunks length for chunks referencing the volume in the CAT table does not equal the `dataleft` value recorded in the VOL entry for the volume.
`dataleft > datawritten`	The entry shows that more data remains on the volume than was written.
`eotchunk < chunksleft`	The entry shows that more chunks remain on the volume than were written.
`eotchunk < largest chunk in cat (443)`	The chunk number of the end-of-volume EOT descriptor on the volume is less than the largest chunk number of all the chunk entries for the volume.

`eotzone < largest zone in cat (77)`	The zone number of the end-of-volume EOT descriptor on the volume is less than the largest zone number of all the chunk entries for the volume.
`missing`	The volume was found in a chunk entry from the CAT table but is not in the VOL table.
`tapesize is bad`	The `tapesize` field for the volume is an impossible number.
`version is bad`	The `version` field for the volume is not 1 or 3 (for a volume still containing data written by the old tape MSP) or 4 (for a volume written by this MSP).
`volume is empty but hfull is on` `volume is empty but hsparse is on`	When a volume is empty, the `hfull` and `hsparse` hold flags should be `off`.

`volume is empty but datawritten != 0`
`volume is empty but eotpos != 1/0`
`volume is empty but eotchunk != 1`
	When the `hfree hold` flag is cleared, the `datawritten` field is set to 0, the `eotpos` field is set to 1/0, and the `eotchunk` is set to 1. The entry is inconsistent and should be checked.

`volume is not empty but hfree is on`	When a volume contains data, the `hfree hold` flag must be `off`.
`volume is not empty and version is` *n* `but hfull is off`	Tapes containing data with a version value of less than 4 must have `hfull` set, because the LS cannot append to the tape.
`volume is not empty and version is` *n* `but eotpos != 2/0`	Tapes imported from the old MSP only have one zone of data, so `eotpos` must be 2/0.
`zonesize is too small`	The `zonesize` field for the volume is an impossible number.

DMF User Library `libdmfusr.so`

The subroutines that constitute the DMF user-command application program interface (API) are available to user-written programs by linking to the DMF user library, `libdmfusr.so`. Sites can design and write their own custom DMF user commands, which eliminates the need to use wrapper scripts around the DMF user commands.

This appendix discusses the following:

- "Overview of the Distributed Command Feature and `libdmfusr.so`" on page 681
- "Considerations for IRIX®" on page 684
- "`libdmfusr.so` Library Versioning" on page 684
- "`libdmfusr.so.2` Data Types" on page 686
- "User-Accessible API Subroutines for `libdmfusr.so.2`" on page 705

Overview of the Distributed Command Feature and `libdmfusr.so`

The distributed command feature allows DMF commands to execute on a host other than the host on which the DMF daemon is running. (This feature was first made available with DMF 2.7.) A host that imports managed filesystems from the DMF daemon host machine can execute the DMF commands locally (see "DMF Manager Web Interface" on page 9). The distributed command feature requires `tcpmux` (RFC 1078).

The DMF user commands communicate with a process named `dmusrcmd`, which is executed as `setuid root`. `dmusrcmd` performs validity checks and communicates with the DMF daemon. (In releases prior to DMF 2.7, user commands communicated directly with the DMF daemon and were installed as `setuid root` processes.)

In order for the DMF user commands to communicate in an efficient and consistent manner with the dmusrcmd process, they must access the DMF user library, which is installed in the following location according to platform operating system and architecture:

Platform	DMF User Library Location
irix-n32	/usr/lib32/libdmfusr.so[.*n*]
irix-64	/usr/lib64/libdmfusr.so[.*n*]
Linux ia64	/usr/lib/libdmfusr.so[.n]

Note: The old version of libdmfusr is located in /usr/lib/dmf/libdmfusr_v1 in order to prevent ldconfig(8) from updating the /usr/lib/libdmfusr.so symbolic link to point to the old library. Customers requiring the version 1 library can make use of it with the following steps:

```
# cd /usr/lib/dmf/libdmfusr_v1
# ln -s libdmfusr.so.1 libdmfusr.so
# export LD_LIBRARY_PATH=/usr/lib/dmf/libdmfusr_v1
```

Linux x86_64	/usr/lib64/libdmfusr.so[.*n*]
Solaris	/usr/lib/sparcv9/libdmfusr.so[.*n*]
Mac OS X	/usr/lib/libdmfusr.[*n*].dylib

Each of the DMF user commands is linked to the library for its protocol-based communications. (The DMF user library became a versioned shared-object library in DMF 3.1. See "libdmfusr.so Library Versioning" on page 684 for more information on accessing the correct version of libdmfusr.so.)

The underlying design of the API calls for the user command to make contact with a dmusrcmd process by creating an opaque context object via a call to the API. This context is then used as a parameter on each function (put, get, fullstat, or copy). The context is used by each API subroutine to perform the requested operation and to correctly return the results of the operation to the command.

In addition to the library, the libdmfusr.H, libdmfcom.H, and dmu_err.h header files are provided. These files are required for sites to effectively create their own commands. All header files are installed in /usr/include/dmf. The libdmf* header files contain all of the object and function prototype definitions required by

the API subroutine calls. The `dmu_err.h` file contains all of the API error code definitions. Along with each error code definition is a text string that is associated with each of the error codes. This text string is the same message that is generated automatically when the error occurs as part of the `DmuErrInfo_t` object (see "`DmuErrInfo_t`" on page 695). The text string is included in the file as informational only, and is not accessible by a program that includes `dmu_err.h`.

Each type of function request (`Put`, `Get`, `Fullstat`, or `Copy`) can be made via a synchronous or an asynchronous API subroutine call:

- Synchronous subroutine calls do not return to the caller until the request has completed, either successfully or unsuccessfully. These synchronous subroutines return an error object to the caller that can be processed to determine the success or failure of the call. If an application is making more than one call, these calls will usually perform less efficiently than their asynchronous counterparts because of the serial nature of their activity.
- Asynchronous subroutine calls return immediately to the caller. The return codes of these asynchronous subroutines indicate whether the request was successfully forwarded to `dmusrcmd` for processing. A successful return allows the calling program to continue its own processing in parallel with the processing being performed by `dmusrcmd` (or the DMF daemon) to complete the request. If the request was successfully forwarded, a request ID that is unique within the scope of the opaque context is returned to the caller. It is the responsibility of the caller to associate the request ID with the correct completion object (described in "`DmuCompletion_t`" on page 692) to determine the eventual result of the original request.

There are several API subroutine calls for processing asynchronous request completion objects. The user can choose to do any of the following:

- Be notified when all requests have completed without processing the return status of each request.
- Process the return status of each request in the order in which they complete. See "`DmuGetNextReply()` Subroutine" on page 730.
- Wait synchronously on an individual asynchronous request's completion by specifying the request ID on which to wait. By using this method, each request return status can be processed in the order in which it was sent, known as *request ID order*.

The API includes well-defined protocols that it uses to communicate with the `dmusrcmd` process. Because these protocols make use of the `pthreads`(5)

mechanism, any user application program making use of the API via libdmfusr.so must also link to the libpthread.so shared object library via one of the following:

-lpthread compiler option using cc(1) or CC(1)
-lpthread loader option using ld(1) or rld(1)

In many cases, the API subroutines pass the address of an object back to the caller by setting a ** pointer accordingly. If errors occur and the subroutine is unable to complete its task, the address returned may be NULL. It is up to the caller to check the validity of an object's address before using it in order to avoid causing a SIGSEGV fault in the application program.

Considerations for IRIX®

The DMF user library for each IRIX platform (lib32 and lib64) was compiled using a MIPSpro™ compiler. Compiling user applications that call DMF user library API subroutines with compilers other than MIPSpro compilers may result in incompatibilities causing load-time or run-time errors.

libdmfusr.so Library Versioning

Caution: The old libdmfusr.so.1 version of the DMF library described below will be removed in a future release. Customers should recompile their applications to use the new library.

DMF 3.1 introduced a new version of the DMF user library. This new version is not compatible with the previous library nor with applications that were written and linked with the previous library. To allow the use of older applications after installing the current version of DMF and to facilitate upgrading older applications, the current version of DMF provides both the old version and the new version and introduces a linking mechanism.

When an application is created and linked with a shared object, the name of the actual library that the application is ultimately linked with is stored in the executable file and used at execution time to find a library of the same name for dynamic linking. In previous releases, the library was named libdmfusr.so. Therefore, all existing DMF commands and site-developed applications that use the library contain the filename libdmfusr.so in the executable for linking with the library at execution time.

A common practice when creating a new version of a library is to add the suffix *.n* to the library name, where *n* is an ever-increasing integer that refers to the current version number.

Prior to DMF 3.1, the library named `libdmfusr.so` was an actual library, rather than a link to a library. The current version of DMF provides the old library (renamed `libdmfusr.so.1`) and the new library (named `libdmfusr.so.2`). All current DMF user commands (such as `dmput`) were created and linked with `libdmfusr.so.2` and their executables contain the filename `libdmfusr.so.2` for linking with the library.

The `libdmfusr.so.1` library is identical to the `libdmfusr.so` library shipped prior to DMF 3.1. The current DMF installation process will install a link named `libdmfusr.so` that will point to `libdmfusr.so.2`. If needed, you can change the link to point to `libdmfusr.so.1` in order to satisfy linking for executables built with a pre-DMF 3.1 `libdmfusr.so`.

The locations of the libraries and the link have not changed from previous releases (see "Overview of the Distributed Command Feature and `libdmfusr.so`" on page 681).

The new `libdmfusr.so` link provides the following advantages:

- You can use the default setting, which does not require any knowledge about the latest version of the library. When developing new site applications using the library, the non-version-specific `ld` option `-ldmfusr` will result in the loader following the link and using the new version of the library, `libdmfusr.so.2`. The resulting applications will contain the name `libdmfusr.so.2` in their executable files for dynamic loading.
- You can reset the link to point to `libdmfusr.so.1`, which allows existing site-developed applications to continue to work with the older version of the library. This will not affect any of the DMF user commands because they contain the name of the new library and make no use of the link at execution time. When an older application executes, if filename `libdmfusr.so` is encountered by the loader and the link points to `libdmfusr.so.1`, the application will continue to work exactly as it did before the current DMF installation.

The two uses of the link as described above are mutually exclusive of each other. Take care when using the link to enable older applications to run with the old library while at the same time developing new applications using the new library. If the link points to `libdmfusr.so.1` and `-ldmfusr` is used to create a new application, the older version of the library will be found and the resulting executable will contain the filename `libdmfusr.so.1` for use at execution time. If older applications are required to run correctly while new applications are being developed, you must use

specific loader command options to ensure that the new applications are linked with the latest library. This can be done by including the specific library name, such as libdmfusr.so.2, on the ld or cc command instead of the generic library specification -ldmfusr.

libdmfusr.so.2 Data Types

The data types described in this section are defined in libdmfusr.H or libdmfcom.H. For the most up-to-date definitions of each of these types, see the appropriate file. The following information is provided as a general description and overall usage outline.

All of the data types defined in this section are C++ objects, and all have constructors and destructors. Many have copy constructors and some have operator override functions defined. Please refer to the appropriate .H header file to see what C++ functions are defined for each object in addition to the member functions described in this section.

DmuAllErrors_t

The DmuAllErrors_t object provides the caller with as much information regarding errors as is practical. The complex nature of the API and its communications allows for many types of errors and several locations (processes) in which they can occur. For example, a request might fail in the API, in the dmusrcmd process, or in the DMF daemon.

The public member fields and functions of this class are as follows:

entry	Specifies a read-only pointer allowing access to all DmuErrInfo_t entries in the DmuAllErrors_t internal array.
numErrors()	Returns the number of DmuErrInfo_t entries in the DmuAllErrors_t internal array.
resetErrors()	Clears the DmuAllErrors_t internal array.

Following is an example using a DmuAllErrors_t object.

Note: The following code is a guideline. It may refer to elements of a `DmuAllErrors_t` structure that are not defined in your installed version of `libdmfcom.H`.

```
report_errors(DmuAllErrors_t  *errs)
{
        int             i;

        if (!errs) {
                return;
        }
        for (i = 0; i < errs->numErrors(); i++) {
                fprintf(stdout, "group '%s' errcode '%d' who '%s' "
                        "severity '%s' position '%s' host '%s' message '%s'\n",
                        errs->entry[i].group ? errs->entry[i].group : "NULL",
                        errs->entry[i].errcode,
                        DmuLogGetErrWhoImage(errs->entry[i].errwho),
                        DmuLogGetSeverityImage(errs->entry[i].severity),
                        errs->entry[i].position ? errs->entry[i].position : "NULL",
                        errs->entry[i].host ? errs->entry[i].host : "NULL",
                        errs->entry[i].message ? errs->entry[i].message : "NULL");
        }
}
```

DmuAttr_t

The `DmuAttr_t` object defines the DMF attribute for a managed file.

The public member fields and functions of this class are as follows:

`bfid` — Specifies a `DmuBfid_t` object (defined in `libdmfcom.H`) that defines the file's bitfile-ID (`bfid`).

`dmflags` — Specifies an integer defining a file's DMAPI flags. Currently unused.

`dmstate` — Specifies a `dmu_state_t` object that defines the file state. Valid states are:

`DMU_ST_ARCHIVING`	Archiving

	DMU_ST_DUALSTATE	Dual-state
	DMU_ST_MIGRATING	Migrating
	DMU_ST_NOMIGR	No migration allowed
	DMU_ST_OFFLINE	Offline
	DMU_ST_PARTIAL	Partial-state file
	DMU_ST_REGULAR	Regular
	DMU_ST_UNMIGRATING	Unmigrating
fsys	Specifies a DmuFileIoMethod_t object (defined in libdmfcom.H) that defines the file's filesystem type.	
regbuf	Specifies a DmuFullRegbuf_t object that defines the file full region information. See "DmuFullRegbuf_t" on page 697.	
sitetag	Defines the file site tag value. See dmtag(1).	
version	Specifies a DmuFileIoVersion_t object (defined in libdmfcom.H) that defines the filesystem version.	

DmuByteRange_t

The DmuByteRange_t object defines a range of bytes that are to be associated with a Put or Get request.

The public member fields and functions of this class are as follows:

start_off	Starting offset in bytes of the range in the file.
end_off	Ending offset in bytes of the range in the file.

Nonnegative values for start_off or end_off indicate an offset from the beginning of the file. The first byte in the file has offset 0. Negative values may be used to indicate an offset from the end of the file. The value -1 indicates the last byte in the file, -2 is the next-to-last byte, and so on. The range is inclusive, so if start_off has a value of 2 and end_off has a value of 2, it indicates a range of one byte.

DmuByteRanges_t

The DmuByteRanges_t object defines a set of DmuByteRange_t objects that are to be associated with a Put or Get request.

The public member fields and functions of this class are as follows:

clearByteRange

Clears the specified byte range in the DmuByteRanges_t object. The clearByteRange() routine is restricted in how it handles negative offsets, both in the DmuByteRange_t members of the DmuByteRanges_t class and in its parameters. The following items give the details of these restrictions. In the following items, *start* and *end* are the parameters to the clearByteRange() routine, using the following format:

clearByteRange(*start*,*end*)

- If *start* and *end* exactly match a DmuByteRange_t entry, then that entry will be cleared. This includes negative numbers.
- If *start* is 0 and *end* is -1, all DmuByteRange_t entries will be cleared. resetByteRanges() is the preferred method for clearing all ranges.
- If *start* is positive and *end* is -1, then:
  - All DmuByteRange_t entrys that have a positive start_off value greater than or equal to *start* will be cleared
  - All DmuByteRange_t entrys that have a positive start_off value that is less than *start* and an end_off value of -1 will be changed to have an end_off value of *start-1* (that is, *start* minus 1). For example, if DmuByteRanges_t has a single range, 3:-1, then clearByteRange(4,-1) will leave a single range, 3:3.
  - All DmuByteRange_t entrys that have a positive start_off value that is less than *start* and an end_off value that is greater than *start* will be changed to have an end_off value of *start-1*. For example, if DmuByteRanges_t has a single range 3:9, then clearByteRange(4,-1) will leave a single range 3:3.

- If *start* and *end* are both positive and a DmuByteRange_t entry has positive start_off and end_off values, then the range specified by *start* and *end* is cleared from the DmuByteRange_t.
- If *start*, *end*, and the start_off and end_off values of a DmuByteRange_t are all negative, the range specified is cleared from DmuByteRange_t.

entry

Specifies a read-only pointer allowing access to all DmuByteRange_t entries in the DmuByteRanges_t internal array.

fromByteRangesImage()

Converts a string that represents a byte range and adds it to the DmuByteRanges_t object. Strings that represent byte ranges are described on the dmput(1) man page.

Note: In a string representing a byte range, -0 represents the last byte in the file; in a DmuByteRange_t object, -1 represents the last byte in the file.

For example, suppose byteranges is declared as the following:

```
DmuByteRanges_t byteranges;
```

Then each of the following statements will add the DmuByteRange_t object that covers the entire file:

```
byteranges.setByteRange(0,-1);
byteranges.fromByteRangesImage("0:-0" ,&errstr);
```

If the byte range overlaps or is adjacent to an existing range in the array, the items may be coalesced.

numByteRanges()

Returns the number of DmuByteRange_t objects contained in the entry array.

resetByteRanges()

Resets the number of DmuByteRange_t objects in the array to zero.

`rounding`

Specifies the rounding method to be used to validate range addresses. Only `DMU_RND_NONE` is valid.

`setByteRange()`

Adds a new range. If the range being added overlaps or is adjacent to an existing range in the array, the items may be coalesced. It is expected that the starting offset not be closer to the end-of-file than the ending offset. For example, a starting offset of `5` and an ending offset of `4` is invalid, and the `setByteRange()` function may not add it to the array. The `setByteRange()` function cannot determine the validity of some ranges, however, and may add ranges that the `Put` or `Get` request will later ignore.

You can create a valid `DmuByteRanges_t` object using the default constructor with or without the `new` operator, depending on the need. For example:

```
DmuByteRanges_t         ranges;

DmuByteRanges_t         *ranges = new DmuByteRanges_t;
```

The following example creates a `DmuByteRanges_t` named `byteranges`, adds a `DmuByteRange_t` to it, then prints the entry to `stdout`:

```
DmuByteRanges_t byteranges;
int             i;
byteranges.rounding = DMU_RND_NONE;
byteranges.setByteRange(0, 4095); /* specifies the first 4096 bytes in the file */
for (i = 0; i < byteranges.numByteRanges(); i++) {
        fprintf(stdout,"Starting offset %lld, ending offset %lld\n",
                byteranges.entry[i].start_off,
                byteranges.entry[i].end_off);
}
```

The output to `stdout` would be as follows:

```
starting offset 0, ending offset 4095
```

The following example creates a `DmuByteRanges_t` named `b`, adds a `DmuByteRange_t` to it, then clears a byte range:

```
DmuByteRanges_t b;
int i;
```

```
b.setByteRange(0,40960);
b.clearByteRange(4096,8191);
printf("Num byte ranges %d\n",b.numByteRanges());
for (i = 0; i < b.numByteRanges(); i++)
        printf("%lld %lld\n",b.entry[i].start_off, b.entry[i].end_off);
```

The output to `stdout` would be as follows:

```
Num byte ranges 2
0 4095
8192 40960
```

Note: The `toByteRangesImage()` member function is not yet supported.

DmuCompletion_t

The `DmuCompletion_t` object is returned by one of the API request completion subroutines (see "Request-Completion Subroutines" on page 727) with the results of an asynchronous request.

The public member fields and functions of this class are as follows:

`fhandle`	Specifies the file handle of the file associated with the request.
`reply_code`	Contains the overall success or failure status of the request. If this value is `DmuNoError`, the request was successful. If not, the output from the `errs` parameter from `DmuGetNextReply()` should be checked for the appropriate error information; see "`DmuGetNextReply()` Subroutine" on page 730.
`request_id`	Associates the completion object with an asynchronous request that was previously issued. This value coincides with the request ID value that any of the asynchronous subroutines return to the user.
`request_type`	Specifies the type of the original request.
`ureq_data`	Specifies a pointer to user request-type specific data. For a `fullstat` user request, this will point to a

DmuFullstat_t object. This field has no meaning for Put, Get, or Copy user requests.

DmuCopyRange_t

The DmuCopyRange_t object defines a range of bytes that are to be associated with a Copy request.

The public member fields and functions of this class are as follows:

dst_offset	Specifies the starting offset in bytes in the destination file to which the copy is sent.
src_length	Specifies the length in bytes of the range to be copied.
src_offset	Specifies the starting offset in bytes of the range in the source file to be copied.

DmuCopyRanges_t

The DmuCopyRanges_t class defines an array of DmuCopyRange_t objects that are to be associated with a Copy request.

The public member fields and functions of this class are as follows:

entry	Specifies a read-only pointer allowing access to all the DmuCopyRange_t entries in the array.
numCopyRanges()	Returns the number of DmuCopyRange_t objects contained in the entry array. Only a single range is supported.
resetCopyRanges()	Resets the number of DmuCopyRange_t objects in the array to zero.
rounding	Specifies the rounding method to be used to validate range addresses. Only DMU_RND_NONE is supported.

setCopyRange	Adds a new DmuCopyRange_t object to the array.

Example: Create a DmuCopyRanges_t, add a DmuCopyRange_t to it, then print the entry to stdout:

```
DmuCopyRanges_t copyranges;
int             i;

copyranges.rounding = DMU_RND_NONE;
copyranges.setCopyRange(0, 4096, 0);

for (i = 0; i < copyranges.numCopyRanges(); i++) {
        fprintf(stdout, "source offset %llu, length %llu, "
                "destination offset %llu\n",
                copyranges.entry[i].src_offset,
                copyranges.entry[i].src_length,
                copyranges.entry[i].dst_offset);
}
```

DmuErrHandler_f

The DmuErrHandler_f object defines a user-specified error handling subroutine. Many of the API subroutines may result in the receipt of error information from the dmusrcmd process or the DMF daemon in the processing of the request. As these errors are received, they are formatted into a DmuErrInfo_t object (see "DmuErrInfo_t" on page 695) and are generally returned to the caller either via a calling parameter or as part of a DmuCompletion_t object.

In addition, however, if the error occurs in the course of processing internal protocol messages, the DmuErrInfo_t object can also be passed into the DmuErrHandler_f that the caller defined when the opaque context was created.

As part of the DmuCreateContext() API subroutine call, the caller can specify a site-defined DmuErrHandler_f subroutine or the caller can use one of the following API-supplied subroutines:

DmuDefErrHandler	Outputs the severity of error and the message associated with the error to stderr.

`DmuNullErrHandler`	Does nothing with the error.

DmuErrInfo_t

The `DmuErrInfo_t` object contains the information about a single error occurrence.

The public member fields and functions of this class are as follows:

`errcode`	Specifies an integer value generated by the originating routine. This code may have many different meanings for a single value, depending on who the originator is.
`errwho`	Specifies an integer value that describes in more detail the originator of the error. Use the `DmuLogGetErrWhoImage()` subroutine to access a character string corresponding to this value.
`group`	Defines the originator of the error: `sgi_dmf` (DMF routine) `sgi_dmf_site` (site-defined policy routine)
`host`	Specifies a character pointer to a string that contains the hostname where the error originated.
`message`	Specifies a character pointer to a string that contains the body of the error message.
`position`	Specifies a character pointer to a string that contains the position of where the error was generated. For example, this could be a pointer to a character string generated using the `__FILE__` and `__LINE__` `cpp`(1) macros. This field may be `NULL`.
`severity`	Specifies an integer value that describes the severity of the error. Use the `DmuLogGetSeverityImage()` subroutine to access a character string corresponding to this value.

DmuError_t

The `DmuError_t` object is the type that most of the API subroutines pass as a return code. The definition `DmuNoError` is the general success return code.

DmuEvents_t

The DmuEvents_t object defines the various event mask settings that a file may contain.

Valid settings are defined as the logical OR of any of the following:

DMF_EVENT_DESTROY	Generates a kernel event for each Destroy request on the file.
DMF_EVENT_READ	Generates a kernel event for each Read request on the file.
DMF_EVENT_TRUNCATE	Generates a kernel event for each Truncate request on the file.
DMF_EVENT_WRITE	Generates a kernel event for each Write request on the file.

DmuFhandle_t

The DmuFhandle_t object contains the ASCII representation of the file fhandle as it is known on the host on which the file's filesystem is native.

The public member fields and functions of this class are as follows:

fromFhandleImage()	Copies an ASCII file handle image string into the hanp field.
hanp	Specifies a character array containing the file handle.
is_valid()	Verifies the validity of the hanp field.
toFhandleImage()	Copies the hanp field into a DmuStringImage_t object.

DmuFsysInfo_t

The DmuFsysInfo_t object contains the subset of DMF filesystem configuration information that may be relevant to a user command.

The public member functions of this class are as follows:

bool supports (DmuOperandType_t op_type);	Returns true if files within the filesystem may be used as operands to a function of op_type. For example, if supports(DmuPutOp) returns true, then it is legal to

	perform a `Put File` request (for example, `DmuPutByPathSync()`) for files in the filesystem; however, individual requests may still fail, depending on configuration parameters. See "`DmuOperandType_t`" on page 700.
`is_configured()`	Returns `true` if the filesystem is defined in the DMF configuration file, either as a managed filesystem or an archive filesystem.
`is_managed()`	Returns `true` if the filesystem is defined in the DMF configuration file and has a `MIGRATION_LEVEL` value other than `archive`. Files in managed filesystems can be used for all `libdmfusr.so` file request subroutines (such as `put` or `get`), with the exception that they cannot be the source file of an `archive` request (`DmuArchiveAsync`/`DmuArchiveSync`).
`is_unmanaged()`	Returns `true` if the filesystem is defined in the DMF configuration file and has a `MIGRATION_LEVEL` value of `archive`. Files in archive filesystems can be used as the source of an `archive` request (`DmuArchiveAsync`/`DmuArchiveSync`) or the destination of a `copy` request (`DmuCopyAsync_2`/`DmuCopySync_2`). (However, see `min_archive_file_size()`.) Archive filesystems do not support `Put`, `Get`, or `Settag` requests, and cannot be used as the source of a `Copy` request.
`min_archive_file_size()`	Specifies the smallest file size that should be submitted in an `archive` request for this filesystem, or a `copy` request when this filesystem is the destination of the `copy` request. This only applies to filesystems for which `is_unmanaged()` is `true`.

`DmuFullRegbuf_t`

The `DmuFullRegbuf_t` object defines the DMF `fullregion` buffer information for a file.

The public member fields and functions of this class are as follows:

`arrcnt`	Specifies the number of regions in the `regions` array.

regcnt	Specifies the number of regions in the regions array that are valid. Only 0 and 1 are supported.
regions	Specifies a DmuFullRegion_t array. See "DmuFullRegion_t" on page 698.

DmuFullRegion_t

The DmuFullRegion_t object defines a file region from address *A* (rg_offset) to address *B* (rg_offset + rg_size) that has a different event mask (described by rg_flags below) from the regions adjacent to it. The object also defines state. When there is more than one region, the file is partial state.

The public member fields and functions of this class are as follows:

rg_fbits	(Reserved for future use.)
rg_flags	Defines the region event flag bitmask. See "DmuEvents_t" on page 696.
rg_offset	Defines the region starting offset in bytes. The start of the file is byte 0.
rg_size	Defines the region size in bytes.
rg_state	Defines the region file state. Valid states are:

DMU_ST_ARCHIVING	Archiving
DMU_ST_DUALSTATE	Dual-state
DMU_ST_MIGRATING	Migrating
DMU_ST_NOMIGR	No migration allowed
DMU_ST_OFFLINE	Offline
DMU_ST_REGULAR	Regular
DMU_ST_UNMIGRATING	Unmigrating

DmuFullstat_t

The DmuFullstat_t object is a user-accessible version of the internal DMF fullstat object. It contains all of the basic stat(2) information regarding the file, as well as all of the DMAPI-related fields.

The public member fields and functions of this class are as follows:

`attr`	Specifies a `DmuAttr_t` object that defines the DMF attribute of the file. See "`DmuAttr_t`" on page 687.
`evmask`	Specifies a `DmuEvents_t` object that defines the event mask for the file. See "`DmuEvents_t`" on page 696.
`host`	Specifies the hostname where the file is native.
`inconsistent`	Indicates that the `DmuFullstat_t` object has inconsistencies in the fields.
`is_valid()`	Returns `1` if the `DmuFullstat_t` is valid.
`mntpt`	Specifies a `DmuOpaque_t` object (defined in `libdmfcom.H`) defining the mount point of the filesystem containing the file on `host`.
`regbuf`	Specifies a `DmuRegionbuf_t` object that defines the regions of the file. See "`DmuRegionbuf_t`" on page 701.
`relpath`	Specifies the relative path of the file in `mntpt` on `host`.
`stat`	Specifies a `DmuStat_t` object that contains the fields representing those in the `stat`(5) structure. See the `stat`(2) system call.

`DmuFullstat_2_t`

The `DmuFullstat_2_t` object is a user-accessible version of the internal DMF `fullstat` object. It contains all of the basic `stat`(2) information regarding the file, as well as all of the DMAPI-related fields.

The public member fields and functions of this class are as follows:

attr	Specifies a DmuAttr_t object that defines the DMF attribute of the file. See "DmuAttr_t" on page 687.
evmask	Specifies a DmuEvents_t object that defines the event mask for the file. See "DmuEvents_t" on page 696.
host	Specifies the hostname where the file is native.
inconsistent	Indicates that the DmuFullstat_2_t object has inconsistencies in the fields.
is_valid()	Returns 1 if the DmuFullstat_2_t is valid.
mntpt	Specifies a DmuOpaque_t object (defined in libdmfcom.H) defining the mount point of the filesystem containing the file on host.
regbuf	Specifies a DmuRegionbuf_t object that defines the regions of the file. See "DmuRegionbuf_t" on page 701.
relpath	Specifies the relative path of the file in mntpt on host.
stat	Specifies a DmuStat_2_t object that contains the fields representing those in the stat(5) structure. See the stat(2) system call.

DmuOperandType_t

The DmuOperandType_t object is an enum data type that is defined in libdmfcom.H and it is used as an operand to DmuFsysInfo_t::supports(). Its values are:

```
DmuPutOp
DmuGetOp
DmuCopySrcOp
DmuCopyDstOp
DmuFstatOp
DmuSettagOp
DmuArchiveSrcOp
DmuArchiveDstOp
DmuSetProjidOp
```

DmuPriority_t

The DmuPriority_t object defines the priority of the request.

Valid settings are defined as follows:

DMU_DEF_PRIORITY	Defines the default priority of the request. You should use DMU_DEF_PRIORITY in any function calls with a deferred priority parameter to guarantee forward compatibility. DMF translates DMU_DEF_PRIORITY to a default value that is within the range DMU_MIN_PRIORITY to DMU_MAX_PRIORITY.
DMU_MIN_PRIORITY	Defines the minimum priority.
DMU_MAX_PRIORITY	Defines the maximum priority.

DmuRegion_t

The DmuRegion_t object defines a file region from address *A* (rg_offset) to address *B* (rg_offset + rg_size) that has a different event mask (described by rg_flags below) from the regions adjacent to it. When there is more than one region, the file is partial state.

The public member fields and functions of this class are as follows:

rg_flags	Defines the region event flag bitmask. See "DmuEvents_t" on page 696.
rg_offset	Defines the region starting offset in bytes. The start of the file is byte 0.
rg_size	Defines the region size in bytes.

DmuRegionbuf_t

The DmuRegionbuf_t object defines the file region buffer information for a file.

The public member fields and functions of this class are as follows:

arrcnt	Specifies the number of regions in the regions array.
regcnt	Specifies the number of regions in the regions array that are valid.

regions	Specifies the DmuRegion_t array. See the DmuRegion_t description.

DmuReplyOrder_t

The DmuReplyOrder_t object selects the order in which asynchronous replies are to be returned by the API reply processing subroutines.

Valid settings are defined as follows:

DmuAnyOrder	Returns replies in the order the replies are received.
DmuReqOrder	Returns replies in the order the requests were issued.

DmuReplyType_t

The DmuReplyType_t object is used to select the type of reply that an API can receive after sending a request. All requests will receive a final reply when the dmusrcmd process has completed processing the request, whether it was successful or not.

Valid settings are defined as follows:

DmuIntermed	Specifies an intermediate reply, an informational message to alert the caller that the request is being processed and may not complete for some time. An example of this is the intermediate reply that is sent when a put request has been forwarded to an MSP or LS for processing and the completion reply is deferred until that operation is complete.
DmuFinal	Specifies the final reply for the request.

This definition is used to specify the types of replies that some of the reply processing subroutines defined below are to consider.

DmuSeverity_t

The DmuSeverity_t object specifies the level of message reporting.

Valid settings are defined as follows:

DmuSevDebug4	Highest level of debug reporting.

DmuSevDebug3	Second-highest level of debug reporting.
DmuSevDebug2	Third-highest level of debug reporting.
DmuSevDebug1	Lowest level of debug reporting.
DmuSevVerbose	Verbose message reporting.
DmuSevInform	Informative message reporting.
DmuSevWarn	Warning message reporting.
DmuSevFatal	Error message reporting.

DmuVolGroup_t

The DmuVolGroup_t object defines a volume group (VG) name. As an entry in a DmuVolGroups_t array, it is used to specify one of the VGs to be used for a DMF put request. For more information about VGs, see "How DMF Software Works" on page 14.

The public member field and function of this class is as follows:

vgname	Specifies a character pointer to a string containing the name of a valid VG.

DmuVolGroups_t

The DmuVolGroups_t object defines an array of DmuVolGroup_t objects. This object is used to specify the list of VGs to which a caller would like a file to be written in a DMF put request.

The public member fields and functions of this class are as follows:

clearVolGroup()	Removes a DmuVolGroup_t object from the internal DmuVolGroup_t array.
fromVolGroupsImage()	Converts a string image of the following format to a DmuVolGroups_t object: *vgname1 vgname2 ...* The delimiter between multiple vgname values may be a space, a tab, or a comma.

numVolGroups()	Returns the number of DmuVolGroup_t objects in the internal DmuVolGroup_t array.
resetVolGroups()	Clears the internal DmuVolGroup_t array.
setVolGroup()	Adds a DmuVolGroup_t object to the internal DmuVolGroup_t array.
toVolGroupsImage()	Converts a DmuVolGroups_t object to a DmuStringImage_t (defined in libdmfcom.H) in the following format: *vgname1 vgname2 ...* The delimiter between multiple vgname values may be a space, a tab, or a comma.

User-Accessible API Subroutines for `libdmfusr.so.2`

This section describes the following types of user-accessible API subroutines:

- "Context-Manipulation Subroutines" on page 705
- "Filesystem-Information Subroutine" on page 708
- "DMF File-Request Subroutines" on page 709
- "Request-Completion Subroutines" on page 727

Context-Manipulation Subroutines

The `DmuContext_t` object manipulated by the `DmuCreateContext()`, `DmuDestroyContext()`, and `DmuChangedDirectory()` subroutines is designed to be completely opaque to the application. The context is used on all API subroutine calls so that the API can successfully manage user request and reply processing, but its internal contents are of no interest or use to the application.

You can use multiple `DmuContext_t` objects within the same process if desired.

`DmuCreateContext()` Subroutine

The `DmuCreateContext()` subroutine creates an opaque context for the API to use to correctly communicate with the `dmusrcmd` process. This subroutine should be the first API subroutine called by a DMF user command. Not only is the context created, but the communication channel to the `dmusrcmd` process is initialized.

Normally, a context would be used for multiple requests and only destroyed when no more requests are to be made. Creating and destroying a context for each request is likely to be inefficient if done frequently.

The prototype is as follows:

```
extern DmuError_t
DmuCreateContext(
        const   char            *prog_name,
                DmuCreateFlags_t create_flags,
                DmuSeverity_t   severity,
                DmuErrHandler_f err_handler,
                DmuContext_t    *dmuctxt,
                pid_t           *child_pid,
```

```
                DmuAllErrors_t  *errs)
```

The parameters are as follows:

`child_pid` — Specifies the process ID (PID) of the child that is forked and executed to create the `dmusrcmd` process. This value is returned to the caller so that the caller is free to handle the termination of child signals as desired.

`create_flags` — Specifies a `DmuCreateFlags_t` object (defined in `libdmfusr.H`) that specifies create options. The only valid `create_flags` option is:

`CREATE_CHDIR`

Allows `change-directory` requests via the `DmuChangedDirectory()` routine. See "`DmuChangedDirectory()` Subroutine" on page 707.

`dmuctxt` — Specifies a `DmuContext_t` object (defined in `libdmfusr.H`) that is returned with the address of the newly created API to be used on all subsequent subroutine calls that require the program's API context.

`err_handler` — Specifies a user-defined error handling subroutine. The `DmuErrHandler_f` object is defined in `libdmfusr.H`. If the `err_handler` parameter is `NULL`, the default error handler `DmuDefErrHandler` is used. For more information, see "`DmuErrHandler_f`" on page 694.

`errs` — Specifies a pointer to a `DmuAllErrors_t` object. This value may be `NULL`. If it is not `NULL`, the subroutine will use it to return errors. See "`DmuAllErrors_t`" on page 686.

`prog_name` — Contains the name of the program. This field can be the full pathname of the program or some other representation.

`severity` — Specifies a `DmuSeverity_t` object that specifies the level of error reporting. See "`DmuSeverity_t`" on page 702.

If the `DmuCreateContext` call completes successfully, it returns `DmuNoError`.

DmuChangedDirectory() Subroutine

The DmuChangedDirectory subroutine changes the current directory of the context. This subroutine is useful to a process that will be making multiple API file requests using relative pathnames while the process might also be making chdir(3) subroutine calls.

When a process makes a chdir call, if the DmuChangedDirectory() subroutine is called before the next API file request that references a relative pathname is made, the file reference will be successfully made by the process.

The prototype is as follows:

```
extern DmuError_t
DmuChangedDirectory(
        const   DmuContext_t    dmuctxt,
        const   char            *new_directory,
        DmuAllErrors_t  *errs);
```

dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
new_directory	Specifies a read-only character pointer to the string containing the directory path that was passed on the last chdir(3) subroutine call.

DmuDestroyContext() Subroutine

The DmuDestroyContext() subroutine destroys the API context dmuctxt. The memory that had been allocated for its use is freed.

The prototype is as follows:

```
extern DmuError_t
DmuDestroyContext(
                DmuContext_t    dmuctxt,
                DmuAllErrors_t  *errs)
```

The parameters are as follows:

dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.

Filesystem-Information Subroutine

The DmuFilesysInfo() routine returns DMF configuration information about a filesystem. The dmarchive(1) command uses this routine to determine whether it can issue an archive or copy request to the DMF daemon when copying data between a source and target.

The DmuFilesysInfo() subroutine does not return until the request has either completed successfully or been aborted due to an error condition.

Upon success, a DmuFsysInfo_t object is transferred to the caller.

The prototype is as follows:

```
DmuError_t
DmuFilesysInfo(
        const   DmuContext_t    dmuctxt,
        const   char            *dmf_path,
        const   char            *fsys_path,
                int             flags,
                DmuFsysInfo_t   *fsys_info,
                DmuAllErrors_t  *errs);
```

The parameters are as follows:

dmf_path	Specifies a path on a managed filesystem, used only for the purposes of locating the DMF server.
dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine

	will use it to return errors. See "DmuAllErrors_t" on page 686.
flags	Specifies one of the flags found in /usr/lib/dmf/libdmfcom.H as FS_INFO*.
fsys_info	Specifies the pointer that will be returned with the DmuFsysInfo_t object.
fsys_path	Specifies a path on the filesystem for which you want configuration information. It does not need to be on a managed filesystem, nor does it need to be a mount point.

If the routine succeeds, it returns DmuNoError.

DMF File-Request Subroutines

Each of the following subroutines makes a DMF file request:

- "Copy File Requests" on page 710
- "Archive File Requests" on page 712
- "Fullstat Requests" on page 714
- "Put File Requests" on page 717
- "Get File Requests" on page 720
- "Setprojid File Requests" on page 723
- "Settag File Requests" on page 725

The context parameter that is included in each of these subroutines must have been already initialized via DmuCreateContext.

Copy File Requests

The DmuCopyAsync_2() and DmuCopySync_2() subroutines perform copy requests in the manner of the dmcopy(1) command. The dmarchive(1) command also issues copy requests when copying from files that are in a migrated state in a managed filesystem.

The DmuCopyAsync_2() subroutine returns immediately after the copy request has been forwarded to the dmusrcmd process. If a reply is desired, the caller must process the reply to this request. See "Request-Completion Subroutines" on page 727.

The DmuCopySync_2() subroutine does not return until the requested copy has either completed successfully or been aborted due to an error condition.

The prototypes are as follows:

```
extern DmuError_t
DmuCopyAsync_2(
        const DmuContext_t    dmuctxt,
        const char            *srcfile_path,
        const char            *dstfile_path,
        const char            *pref_vgmg,
              DmuCopyFlags_t  copy_flags,
        const DmuCopyRanges_t *copyranges,
              DmuPriority_t   priority,
              DmuReqid_t      *request_id,
              DmuAllErrors_t  *errs)

extern DmuError_t
DmuCopySync_2(
        const DmuContext_t    dmuctxt,
        const char            *srcfile_path,
        const char            *dstfile_path,
        const char            *pref_vgmg,
              DmuCopyFlags_t  copy_flags,
        const DmuCopyRanges_t *copyranges,
              DmuPriority_t   priority,
              DmuAllErrors_t  *errs)
```

The parameters are as follows:

copy_flags	Specifies the OR'd value of the following copy operation flags as defined in libdmfcom.H:

- COPY_CANCEL — Cancel matching request
- COPY_ADDR_ALIGN — Allow an address in the destination file that is greater than the size of the file
- COPY_NONE — No flags specified
- COPY_NOWAIT — Return immediately if the daemon is not available to process the request (do not wait)
- COPY_PRESV_DFILE — Do not truncate the destination file before the copy operation
- COPY_REPRI — Reprioritize matching request

Note: The following flags may only be set by the root user on the DMF server or on a *VALID_ROOT_HOSTS* server:

COPY_CANCEL
COPY_REPRI

See "base Object Parameters" on page 278.

---

copyranges	Specifies a pointer to a DmuCopyRanges_t object, as defined in "DmuCopyRanges_t" on page 693 and in libdmfcom.H. This object can have only one DmuCopyRange_t as defined in "DmuCopyRange_t" on page 693 and in libdmfcom.H.
dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
dstfile_path	Specifies the pathname of the destination (output) file for the copy operation. This path must point to a file that exists or can be created on a filesystem visible from the DMF server and any parallel data-mover nodes. See also "DMF Direct Archiving Requirements" on page 50.
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.

pref_vgmg	Specifies a list of MSP, volume group, or migrate group names from which the file should be copied if possible. If the file has a copy in one of the elements in the list, the DMF daemon will attempt to copy the file first using that copy, overriding the default copy order as defined in the DMF configuration file. This option is only available to the root user.
priority	Specifies a DmuPriority_t object (defined in libdmfcom.H) that defines the request priority.
request_id	Specifies a pointer to a DmuReqid_t object (defined in libdmfcom.H) parameter that will be returned with the unique request ID of the asynchronous request. This value can be used when processing DmuCompletion_t objects (see "Request-Completion Subroutines" on page 727).
srcfile_path	Specifies the pathname of the source (input) file for the copy operation. It must be an offline or dual-state DMF file.

If the subroutine succeeds, it returns DmuNoError.

Archive File Requests

The DmuArchiveAsync() and DmuArchiveSync() subroutines perform archive requests in the manner of the dmarchive(1) command, when dmarchive is operating in the mode of copying files from an archive filesystem to a managed filesystem.

The DmuArchiveAsync() subroutine returns immediately after the archive request has been forwarded to the dmusrcmd process. If a reply is desired, the caller must process the reply to this request. See "Request-Completion Subroutines" on page 727.

The DmuArchiveSync() subroutine does not return until the requested archive has either completed successfully or been aborted due to an error condition.

The prototypes are as follows:

```
extern DmuError_t
DmuArchiveAsync(
        const   DmuContext_t    dmuctxt,
        const   char            *src_path,
```

```
        const   char            *dst_path,
        const   DmuVolGroups_t  *volgroups,
                int             arch_flags,
                DmuPriority_t   priority,
                DmuReqid_t      *request_id,
                DmuAllErrors_t  *errs);

extern DmuError_t
DmuArchiveSync(
        const   DmuContext_t    dmuctxt,
        const   char            *src_path,
        const   char            *dst_path,
        const   DmuVolGroups_t  *volgroups,
                int             arch_flags,
                DmuPriority_t   priority,
                DmuAllErrors_t  *errs);
```

The parameters are as follows:

`arch_flags` — Specifies the OR'd value of the following `archive` operation flags as defined in `libdmfcom.H`:

- `ARCH_CANCEL` — Cancel matching request
- `ARCH_NONE` — No flags specified
- `ARCH_NOWAIT` — Return immediately if the daemon is not available to process the request (do not wait)

Note: The `ARCH_CANCEL` flag may only be set by the `root` user on the DMF server or on a *VALID_ROOT_HOSTS* server. See "`base` Object Parameters" on page 278.

`dmuctxt` — Specifies a `DmuContext_t` object that was previously created by `DmuCreateContext()`.

`dst_path` — Specifies the pathname of the destination (output) file for the `archive` operation. This path must refer to a file on a managed filesystem that either does not currently exist or exists and is zero-length.

errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
priority	Specifies a DmuPriority_t object (defined in libdmfcom.H) that defines the request priority. (Deferred implementation.)
request_id	Specifies a pointer to a DmuReqid_t object (defined in libdmfcom.H) parameter that will be returned with the unique request ID of the asynchronous request. This value can be used when processing DmuCompletion_t objects (see "Request-Completion Subroutines" on page 727).
src_path	Specifies the pathname of the source (input) file for the archive operation. It must be a file in a non-DMF managed filesystem. See "DMF Direct Archiving Requirements" on page 50.
volgroups	Specifies a pointer to a DmuVolGroups_t object. See "DmuVolGroups_t" on page 703.

If the subroutine succeeds, it returns DmuNoError.

Fullstat Requests

The following subroutines send a fullstat request to the dmusrcmd process:

- Current subroutines (as of DMF 6.2)

  ```
  DmuFullstatByFhandleAsync_2()
  DmuFullstatByFhandleSync_2()
  DmuFullstatByPathAsync_2()
  DmuFullstatByPathSync_2()
  ```

- Legacy subroutines (prior to DMF 6.2):

  ```
  DmuFullstatByFhandleAsync()
  DmuFullstatByFhandleSync()
  DmuFullstatByPathAsync()
  DmuFullstatByPathSync()
  ```

Note: If your program must work on versions of DMF prior to DMF 6.2, you should use the legacy versions of the routines. However, the project id field is only available in the new (`Dmu*_2`) versions of the routines.

These subroutines have the following things in common:

- The '`Sync`' versions of these subroutines do not return until the `DmuFullstat_t` has been received or the request has been aborted due to errors.
- The '`Async`' versions of these subroutines return immediately after successfully forwarding the `fullstat` request to the `dmusrcmd` process. If a reply is desired, the caller must process the reply to this request. See "Request-Completion Subroutines" on page 727. That is the only way to actually receive the `DmuFullstat_t` object for an '`Async`' `fullstat` request, however. The `DmuFullstatCompletion()` subroutine has been supplied to extract the `fullstat` information from a `fullstat` completion object.
- The '`ByPath`' versions of these subroutines allow the target file to be defined by its pathname.
- The '`ByFhandle`' versions of these subroutines allow the target file to be defined by its filesystem handle, the `fhandle`. These subroutines are valid only when the command making the call is on the DMF server machine, and they are valid only when a user has sufficient (`root`) privileges.

These subroutines can return a successful completion (`DmuNoError`), but might not return valid `DmuFullstat_t` information. The subroutines are designed to return the normal `stat` type information regardless of whether a DMAPI `fullstat` could be successfully completed. Upon return from these subroutines, the caller can use the `DmuFullstat_t is_valid()` member function to verify the validity of the DMAPI information in the `DmuFullstat_t` block.

The ultimate result of this request is the transfer of a `DmuFullstat_t` object to the caller.

The prototypes are as follows:

```
extern DmuError_t
DmuFullstatByFhandleAsync(
                const    DmuContext_t   dmuctxt,
                const    DmuFhandle_t   *client_fhandle,
                         DmuReqid_t     *request_id,
                         DmuAllErrors_t *errs)

extern DmuError_t
DmuFullstatByFhandleSync(
                const    DmuContext_t   dmuctxt,
                const    DmuFhandle_t   *client_fhandle,
                         DmuFullstat_t  *dmufullstat,
                         DmuAllErrors_t *errs)

extern DmuError_t
DmuFullstatByPathAsync(
                const    DmuContext_t   dmuctxt,
                const    char           *path,
                         DmuReqid_t     *request_id,
                         DmuAllErrors_t *errs)

extern DmuError_t
DmuFullstatByPathSync(
                const    DmuContext_t   dmuctxt,
                const    char           *path,
                         DmuFullstat_t  *dmufullstat,
                         DmuFhandle_t   *fhandle,
                         DmuAllErrors_t *errs)
```

The parameters are as follows:

`dmuctxt`	Specifies a `DmuContext_t` object that was previously created by `DmuCreateContext()`.
`dmufullstat`	Specifies the pointer that will be returned with the `DmuFullstat_t` object.
`client_fhandle`	Specifies the DMF filesystem `fhandle` of the target file. Valid for use only by a privileged (`root`) user on the DMF server machine.

errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
fhandle	Specifies the pointer that will be returned with the DmuFhandle_t value.
path	Specifies the relative or absolute pathname of the target file.
request_id	Specifies a pointer to a DmuReqid_t object (defined in libdmfcom.H) parameter that will be returned with the unique request ID of the asynchronous request. This value can be used when processing DmuCompletion_t objects (see "Request-Completion Subroutines" on page 727).

If the subroutine succeeds, it returns DmuNoError.

Put File Requests

The following subroutines perform the put DMF request:

```
DmuPutByFhandleAsync()
DmuPutByFhandleSync()
DmuPutByPathAsync()
DmuPutByPathSync()
```

These subroutines have the following things in common:

- The 'Sync' versions do not return until the put request has either completed successfully, or been aborted due to errors.
- The 'Async' versions return immediately after successfully forwarding the put request to the dmusrcmd process. If a reply is desired, the caller must process the reply to this request. See "Request-Completion Subroutines" on page 727.
- The 'ByPath' versions allow the target file to be defined by its pathname.
- The 'ByFhandle' versions allow the target file to be defined by its filesystem handle, the fhandle. These subroutines are valid only when the command making the call is on the DMF server machine, and they are valid only when a user has sufficient (root) privileges.

The prototypes are as follows:

```
extern DmuError_t
DmuPutByFhandleAsync(
               const   DmuContext_t    dmuctxt,
               const   DmuFhandle_t    *client_fhandle,
                       DmuMigFlags_t   mig_flags,
               const   DmuByteRanges_t *byteranges,
               const   DmuVolGroups_t  *volgroups,
                       DmuPriority_t   priority,
                       DmuReqid_t      *request_id,
                       DmuAllErrors_t  *errs)

extern DmuError_t
DmuPutByFhandleSync(
               const   DmuContext_t    dmuctxt,
               const   DmuFhandle_t    *client_fhandle,
                       DmuMigFlags_t   mig_flags,
               const   DmuByteRanges_t *byteranges,
               const   DmuVolGroups_t  *volgroups,
                       DmuPriority_t   priority,
                       DmuAllErrors_t  *errs)

extern DmuError_t
DmuPutByPathAsync(
               const   DmuContext_t    dmuctxt,
               const   char            *path,
                       DmuMigFlags_t   mig_flags,
               const   DmuByteRanges_t *byteranges,
               const   DmuVolGroups_t  *volgroups,
                       DmuPriority_t   priority,
                       DmuReqid_t      *request_id,
                       DmuAllErrors_t  *errs)

extern DmuError_t
DmuPutByPathSync(
               const   DmuContext_t    dmuctxt,
               const   char            *path,
                       DmuMigFlags_t   mig_flags,
               const   DmuByteRanges_t *byteranges,
               const   DmuVolGroups_t  *volgroups,
```

```
                    DmuPriority_t   priority,
                    DmuAllErrors_t  *errs)
```

The parameters are as follows:

byteranges	Specifies a pointer to a DmuByteRanges_t object. See "DmuByteRanges_t" on page 689.
client_fhandle	Specifies the DMF filesystem fhandle of the target file. Valid for use only by a privileged (root) user on the DMF server machine.
dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
mig_flags	Specifies the following migration flags as defined in libdmfcom.H: • MIG_CANCEL — Cancel matching request • MIG_FREE — Free the space associated with the file • MIG_NONE — No flags specified • MIG_NOWAIT — Return immediately if the daemon is not available to process the request (do not wait)
	Note: The MIG_CANCEL flag may only be set by the root user on the DMF server or on a *VALID_ROOT_HOSTS* server. See "base Object Parameters" on page 278.
path	Specifies the relative or full pathname of the target file.
priority	Specifies a DmuPriority_t object (defined in libdmfcom.H) that defines the request priority. (Deferred implementation.)
request_id	Specifies a pointer to a DmuReqid_t object (defined in libdmfcom.H) parameter that will be returned with

	the unique request ID of the asynchronous request. This value can be used when processing DmuCompletion_t objects (see "Request-Completion Subroutines" on page 727).
volgroups	Specifies a pointer to a DmuVolGroups_t object. See "DmuVolGroups_t" on page 703.

If the subroutine succeeds, it returns DmuNoError.

Get File Requests

The following subroutines perform the get DMF request:

```
DmuGetByFhandleAsync_2()
DmuGetByFhandleSync_2()
DmuGetByPathAsync_2()
DmuGetByPathSync_2()
```

These subroutines have the following things in common:

- The 'Sync' versions do not return until the get request has either completed successfully or has been aborted due to errors.
- The 'Async' versions return immediately after successfully forwarding the get request to the dmusrcmd process. If a reply is desired, the caller must process the reply to this request. See "Request-Completion Subroutines" on page 727.
- The 'ByPath' versions of these calls allow the target file to be defined by its pathname.
- The 'ByFhandle' versions allow the target file to be defined by its filesystem handle, the fhandle. These subroutines are valid only when the command making the call is on the DMF server machine, and they are valid only when a user has sufficient (root) privileges.

The prototypes are as follows:

```
extern DmuError_t
DmuGetByFhandleAsync_2(
        const   DmuContext_t    dmuctxt,
        const   DmuFhandle_t    *client_fhandle,
                int             recall_flags,
        const   DmuByteRanges_t *byteranges,           /* may be NULL */
```

```
        const   char            *pref_vgmg,             /* may be NULL */
                DmuPriority_t   priority,
                DmuReqid_t      *request_id,
                DmuAllErrors_t  *errs);

extern DmuError_t
DmuGetByFhandleSync_2(
        const   DmuContext_t    dmuctxt,
        const   DmuFhandle_t    *client_fhandle,
                int             recall_flags,
        const   DmuByteRanges_t *byteranges,            /* may be NULL */
        const   char            *pref_vgmg,             /* may be NULL */
                DmuPriority_t   priority,
                DmuAllErrors_t  *errs);

extern DmuError_t
DmuGetByPathAsync_2(
        const   DmuContext_t    dmuctxt,
        const   char            *path,
                int             recall_flags,
        const   DmuByteRanges_t *byteranges,            /* may be NULL */
        const   char            *pref_vgmg,             /* may be NULL */
                DmuPriority_t   priority,
                DmuReqid_t      *request_id,
                DmuAllErrors_t  *errs);

extern DmuError_t
DmuGetByPathSync_2(
        const   DmuContext_t    dmuctxt,
        const   char            *path,
                int             recall_flags,
        const   DmuByteRanges_t *byteranges,            /* may be NULL */
        const   char            *pref_vgmg,             /* may be NULL */
                DmuPriority_t   priority,
                DmuAllErrors_t  *errs);
```

The parameters are as follows:

`byteranges` Specifies a pointer to a `DmuByteRanges_t` object. See "`DmuByteRanges_t`" on page 689.

client_fhandle	Specifies the DMF filesystem fhandle of the target file. Valid for use only by a privileged (root) user on the DMF server machine.
dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
path	Specifies the relative or full pathname of the target file.
pref_vgmg	Specifies a list of MSP, volume group, or migrate group names from which the files should be recalled if possible. If a file has a copy in one of the elements in the list, the DMF daemon will attempt to recall the file first using that copy, overriding the default recall order as defined in the DMF configuration file. This option is only available to the root user.
priority	Specifies a DmuPriority_t object (defined in libdmfcom.H) that defines the request priority.
recall_flags	Specifies the following recall flags as defined in libdmfcom.H: • RECALL_ATIME — Update the access time of the file. This parameter is only valid with DmuGetByPathAsync_2() and DmuGetByPathSync_2(). • RECALL_CANCEL — Cancel matching request • RECALL_NONE — No flags specified • RECALL_NOWAIT — Return immediately if the daemon is not available to process the request (do not wait) • RECALL_REPRI — Reprioritize matching request

Note: The following flags may only be set by the `root` user on the DMF server or on a *VALID_ROOT_HOSTS* server:

```
RECALL_CANCEL
RECALL_REPRI
```

See "`base` Object Parameters" on page 278.

`request_id`	Specifies a pointer to a `DmuReqid_t` (defined in `libdmfcom.H`) parameter that will be returned with the unique request ID of the asynchronous request. This value can be used when processing `DmuCompletion_t` objects (see "Request-Completion Subroutines" on page 727).

If the subroutine succeeds, it returns `DmuNoError`.

`Setprojid` File Requests

The `Setprojid` request performs the same functional task as the `dmprojid`(1) command. The following subroutines perform the `Setprojid` DMF request:

```
DmuSetprojidByPathAsync()
DmuSetprojidByPathSync()
```

Note the following:

- The '`Sync`' version does not return until the `Setprojid` request has either completed successfully or has been aborted due to errors.
- The '`Async`' version returns immediately after successfully forwarding the `Setprojid` request to the `dmusrcmd` process. If a reply is desired, the caller must process the reply to this request. See "Request-Completion Subroutines" on page 727.
- They allow the target file to be defined by its pathname.

The prototypes are as follows:

```
extern DmuError_t
DmuSetprojidByPathAsync(
        const   DmuContext_t    dmuctxt,
```

```
        const   char            *path,
                int             setprojid_flags,
                uint32_t        projid,
                DmuPriority_t   priority,               /* deferred */
                DmuReqid_t      *request_id,
                DmuAllErrors_t  *errs);                 /* may be NULL */

extern DmuError_t
DmuSetprojidByPathSync(
        const   DmuContext_t    dmuctxt,
        const   char            *path,
                int             setprojid_flags,
                uint32_t        projid,
                DmuPriority_t   priority,               /* deferred */
                DmuAllErrors_t  *errs);                 /* may be NULL */
```

The parameters are as follows:

dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
path	Specifies the relative or full pathname of the target file.
priority	Specifies a DmuPriority_t object (defined in libdmfcom.H) that defines the request priority. (Deferred implementation.)
projid	Specifies the file project ID value. See the dmprojid(8) man page.
request_id	Specifies a pointer to a DmuReqid_t (defined in libdmfcom.H) parameter that will be returned with the unique request ID of the asynchronous request. This value can be used when processing DmuCompletion_t objects (see "Request-Completion Subroutines" on page 727).
setprojid_flags	Specifies the following flags:

- PROJID_CLEARINHERIT — Clear the project ID inheritance bit on directories
- PROJID_NONE — No flags specified
- PROJID_NOWAIT — Do not wait if the DMF daemon is unavailable
- PROJID_SETINHERIT — Set the project ID inheritance bit on directories

Note: These flags may only be set by the root user on the DMF server or on a *VALID_ROOT_HOSTS* server (see "base Object Parameters" on page 278).

If the subroutine succeeds, it returns DmuNoError.

Settag File Requests

The Settag request performs the same functional task as the dmtag(1) command. The following subroutines perform the Settag DMF request:

```
DmuSettagByFhandleAsync()
DmuSettagByFhandleSync()
DmuSettagByPathAsync()
DmuSettagByPathSync()
```

These subroutines have the following things in common:

- The 'Sync' versions do not return until the Settag request has either completed successfully or has been aborted due to errors.
- The 'Async' versions return immediately after successfully forwarding the Settag request to the dmusrcmd process. If a reply is desired, the caller must process the reply to this request. See "Request-Completion Subroutines" on page 727.
- The 'ByPath' versions allow the target file to be defined by its pathname.
- The 'ByFhandle' versions allow the target file to be defined by its filesystem handle, the fhandle. These subroutines are valid only when the command making the call is on the DMF server machine and when a user has sufficient (root) privileges.

The prototypes are as follows:

```
extern DmuError_t
DmuSettagByFhandleAsync(
               const   DmuContext_t     dmuctxt,
               const   DmuFhandle_t     *client_fhandle,
                       DmuSettagFlags_t settag_flags,
                       DmuSitetag_t     sitetag,
                       DmuPriority_t    priority,
                       DmuReqid_t       *request_id,
                       DmuAllErrors_t   *errs)

extern DmuError_t
DmuSettagByFhandleSync(
               const   DmuContext_t     dmuctxt,
               const   DmuFhandle_t     *client_fhandle,
                       DmuSettagFlags_t settag_flags,
                       DmuSitetag_t     sitetag,
                       DmuPriority_t    priority,
                       DmuAllErrors_t   *errs)

extern DmuError_t
DmuSettagByPathAsync(
               const   DmuContext_t     dmuctxt,
               const   char             *path,
                       DmuSettagFlags_t settag_flags,
                       DmuSitetag_t     sitetag,
                       DmuPriority_t    priority,
                       DmuReqid_t       *request_id,
                       DmuAllErrors_t   *errs)

extern DmuError_t
DmuSettagByPathSync(
               const   DmuContext_t     dmuctxt,
               const   char             *path,
                       DmuSettagFlags_t settag_flags,
                       DmuSitetag_t     sitetag,
                       DmuPriority_t    priority,
                       DmuAllErrors_t   *errs)
```

The parameters are as follows:

client_fhandle	Specifies the DMF filesystem fhandle of the target file. Valid for use only by a privileged (root) user on the DMF server machine.
dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
path	Specifies the relative or full pathname of the target file.
priority	Specifies a DmuPriority_t object (defined in libdmfcom.H) that defines the request priority. (Deferred implementation.)
request_id	Specifies a pointer to a DmuReqid_t (defined in libdmfcom.H) parameter that will be returned with the unique request ID of the asynchronous request. This value can be used when processing DmuCompletion_t objects (see "Request-Completion Subroutines" on page 727).
settag_flags	Specifies the following flags as defined in libdmfcom.H: • SETTAG_NONE — No flags specified • SETTAG_NOWAIT — Return immediately if the daemon is not available to process the request (do not wait)
sitetag	Defines the file site tag value. See dmtag(1).

If the subroutine succeeds, it returns DmuNoError.

Request-Completion Subroutines

The request completion subroutines are provided so that the application can process the completion events of any asynchronous requests it might have issued. The caller can choose to process each request's completion object (DmuCompletion_t) or to be notified when each request has responded with either an intermediate or final (completion) reply.

The asynchronous requests described previously along with the following completion subroutines allow the user to achieve maximum parallelization of the processing of all requests.

DmuAwaitReplies() Subroutine

The DmuAwaitReplies() subroutine performs a synchronous wait until the number of outstanding request replies of the type specified is less than or equal to max_outstanding. This subroutine is called by a user who does not want to perform individual processing of each outstanding request, but wants to know when a reply (intermediate or final) has been received for each request that has been sent to this point.

The prototype is as follows:

```
extern DmuError_t
DmuAwaitReplies(
        const   DmuContext_t    dmuctxt,
                DmuReplyType_t  type,
                int             max_outstanding,
                DmuAllErrors_t  *errs)
```

The parameters are as follows:

dmuctxt	Specifies a DmuContext_t object that was previously created by DmuCreateContext().
errs	Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.
max_outstanding	Specifies the number of outstanding requests allowed for which the type reply has not been received before the subroutine returns. If this parameter is 0, all type replies will have been received when the subroutine returns.
type	Defines the type of reply to be received. The caller can wait for an intermediate or final reply for the outstanding requests.

See the definition of DmuReplyType_t in "DmuReplyType_t" on page 702 or in libdmfcom.H.

If no errors occurred getting the next reply, this subroutine returns DmuNoError.

DmuFullstatCompletion() Subroutine

The DmuFullstatCompletion() subroutine can be called when asynchronous fullstat replies are being processed by DmuGetNextReply() or DmuGetThisReply(). When the reply is received, the DmuCompletion_t object that is part of the reply can be used as an input parameter to this routine, which will then extract the DmuFullstat_t object and the DmuFhandle_t objects that are contained in the DmuCompletion_t object's ureq_data field.

The prototype is as follows:

```
extern DmuError_t
DmuFullstatCompletion(
                DmuCompletion_t  *comp;
                DmuFullstat_t    *dmufullstat,
                DmuFhandle_t     *fhandle,
                DmuAllErrors_t   *errs)
```

The parameters are as follows:

comp — Specifies the DmuCompletion_t object from an asynchronous Fullstat request.

dmufullstat — Specifies a pointer to an existing DmuFullstat_t object. If comp references a successful Fullstat request, dmufullstat will be set to be equal to the DmuFullstat_t that was returned with the reply.

errs — Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.

fhandle — Specifies a pointer to an existing DmuFhandle_t object. If comp references a successful Fullstat request,

fhandle will be set to be equal to the DmuFhandle_t that was returned with the reply.

DmuGetNextReply() Subroutine

The DmuGetNextReply() subroutine returns the completion object of the next reply based on the order specified on the call. This subroutine performs a synchronous wait until a request reply of the type specified on the call is received. At the time of the call, any reply that has already been received and is queued for processing is returned immediately.

The prototype is as follows:

```
extern DmuError_t
DmuGetNextReply(
        const   DmuContext_t    dmuctxt,
                DmuReplyOrder_t order,
                DmuReplyType_t  type,
                DmuCompletion_t *comp,
                DmuAllErrors_t  *errs)
```

The parameters are as follows:

comp — Specifies a pointer to an existing DmuCompletion_t object. If a reply was available for processing according to the parameters on the calling subroutine, the DmuCompletion_t object pointed to by comp will be set with all of the appropriate values. See "DmuCompletion_t" on page 692.

dmuctxt — Specifies a DmuContext_t object that was previously created by DmuCreateContext().

errs — Specifies a pointer to a DmuAllErrors_t object. This value may be NULL. If it is not NULL, the subroutine will use it to return errors. See "DmuAllErrors_t" on page 686.

Note: This object will return errors that occurred while waiting for or receiving this reply. It does not refer to the errors that might have occurred during the request processing that resulted in the reply. Those errors are available in the comp object.

`order` — Defines the order in which the request replies should be returned. The caller can process the replies in the order the replies are received (`DmuAnyOrder`) or in the order the requests were issued (`DmuReqOrder`).

See the definition of `DmuReplyOrder_t` in "`DmuReplyOrder_t`" on page 702 or in `libdmfcom.H`.

`type` — Defines the type of reply to be received, which is one of the following:

- `DmuIntermed` returns both intermediate messages (which indicate that the DMF daemon has received a valid request) and the final message (which indicates that the request is complete, either with a successful execution or with errors).

  The `comp.ureq_data` field will aways be `NULL` for an intermediate reply and for failed final replies. For successful replies, there are various possible structures returned via `ureq_data`, or `NULL`.

- `DmuFinal` returns only the final message (intermediate messages are silently discarded)

For example:

```
rc = DmuGetNextReply(ctxp, DmuAnyOrder, DmuIntermed, &comp, &errs);
 if (rc != DmuNoError) {
         /* The request could not be sent,  this request is complete. */
 } else if (comp.reply_code == /* intermediate-reply */ 1) {
         /* This is an intermediate reply, the request is accepted, a final reply will come later. */
 } else if (comp.reply_code != DmuNoError) {
         /* This is a failed final reply, DMF could not complete your request. */
 } else {
         /* This is a successful final reply, the request is complete. */
 }
```

The value of `comp.ureq_data` in a successful final reply depends upon which request type was sent (a few request types return data):

- A pointer to a DmuFullstat_t structure will be returned for the following request types:

  DmuFullstatByFhandleAsync
  DmuFullstatByFhandleSync
  DmuFullstatByPathAsync
  DmuFullstatByPathSync

- A pointer to a DmuFullstat_2_t structure will be returned for the following request types:

  DmuFullstatByFhandleAsync_2
  DmuFullstatByFhandleSync_2
  DmuFullstatByPathAsync_2
  DmuFullstatByPathSync_2

- A pointer to a DmuFsysInfo_t structure will be returned for the following request type:

  DmuFilesysInfo

- All other request types return a value of NULL.

If no errors occurred getting the next reply, this subroutine returns DmuNoError. If there are no outstanding requests pending, a return code of DME_DMU_QUEUEEMPTY is returned. You can use a check for DME_DMU_QUEUEEMPTY to terminate a while loop based on this subroutine. Any other error return code indicates an error, and the errs parameter can be processed for the error information.

DmuGetThisReply() Subroutine

The DmuGetThisReply() subroutine returns the completion object of the specified request. This subroutine performs a synchronous wait until a request reply specified on the call is received.

The prototype is as follows:

```
extern DmuError_t
DmuGetThisReply(
        const   DmuContext_t    dmuctxt,
                DmuReqid_t      request_id,
                DmuCompletion_t *comp,
                DmuAllErrors_t  *errs)
```

The parameters are as follows:

`comp`
: Specifies a pointer to an existing `DmuCompletion_t` object. If a reply was available for processing according to the parameters on the calling subroutine, the `DmuCompletion_t` object pointed to by `comp` will be set with all of the appropriate values. See "`DmuCompletion_t`" on page 692.

  The `reply_code` field of the `comp` parameter is the ultimate status of the request. A successful `comp` has a `reply_code` of `DmuNoError`. If the `reply_code` of `comp` is not `DmNoError`, the `comp->allerrors` object will contain the error information needed to determine the cause of the error.

  Note: The `errs` parameter on the subroutine call does not contain the error information for the failed request.

`dmuctxt`
: Specifies a `DmuContext_t` object that was previously created by `DmuCreateContext()`.

`errs`
: Specifies a pointer to a `DmuAllErrors_t` object. This value may be `NULL`. If it is not `NULL`, the subroutine will use it to return errors. See "`DmuAllErrors_t`" on page 686.

  Note: This object will return errors that occurred while waiting for or receiving this reply. It does not refer to the errors that might have occurred during the request processing that resulted in the reply. Those errors are available in the `comp` object.

`request_id`
: Specifies the unique request ID of the request for which the caller wants to wait.

If no errors occurred getting the next reply, this subroutine returns `DmuNoError`. Any other error return code indicates an error, and the `errs` parameter can be processed for the error information.

Site-Defined Policy Subroutines and the `sitelib.so` Library

This appendix provides an overview of the site-defined policy feature and a summary of the policy subroutines sites may write:

- "Overview of Site-Defined Policy Subroutines" on page 735
- "Getting Started with Custom Subroutines" on page 736
- "Considerations for Writing Custom Subroutines" on page 738
- "`sitelib.so` Data Types" on page 739
- "Site-Defined Policy Subroutines" on page 743
- "Helper Subroutines for `sitelib.so`" on page 752

Overview of Site-Defined Policy Subroutines

Site-defined policy subroutines are loaded dynamically by DMF to provide custom decision-making at key points in its processing. Several DMF processes, including `dmfdaemon`, can call subroutines within `sitelib.so`.

You do not need to use this feature, in which case DMF will function as documented in the manuals and man pages. But if you wish, you can implement one or more of these subroutines in order to override DMF's default behavior.

If you use the site-defined policy feature, you must communicate the policy changes to your user community; otherwise, they will not be able to predict how the user commands will work. The man page for any command with a site-defined policy will state something like the following:

> If your site is using the site-defined policy feature, the default behavior may be overridden. Please check with your administrator for any behavior differences due to site-defined policies.

You should also consider adding `ERROR`, `WARN`, and `INFO` messages into the reply stream for commands you customize so that you can routinely return messages to the user that explain what was changed in their request. Doing so will allows the users to understand why the behavior was different from what they expected.

The subroutines are written in C++ according to the subroutine prototypes in /usr/include/dmf/libdmfadm.H. They are placed in a shared-object library called /usr/lib/dmf/sitelib.so.

The parameters and return values of the subroutines and the name of the sitelib.so library are fixed and cannot be altered by the site. In general, the parameters provide all of the information DMF has that is relevant to the purpose of the subroutine, which is described in the comments preceding each subroutine.

The code within the subroutines performs whatever processing the site wishes. To assist in several common operations, such as extracting information from the DMF configuration file, optional helper subroutines are provided in /usr/include/dmf/libdmfadm.H.

Getting Started with Custom Subroutines

The /usr/share/doc/dmf-*/info/sample directory contains the following files to demonstrate generating the sitelib.so library:

- sample_sitelib.C contains source code of basic sample subroutines
- sample_sitelib.mk is the associated makefile

Note: If you use these files as a base for implementing subroutines of your own, be sure to keep them in a different directory and/or rename them to avoid any conflict when DMF is upgraded and new sample files are installed. For example, you could rename the files sitelib.c and sitelib.mk.

For example, to use the basic subroutine example `sample_sitelib.C`, do the following:

1. Copy `sample_sitelib.C` and its associated makefile `sample_sitelib.mk` from `/usr/share/doc/dmf-*/info/sample` to a directory of your own with names of your own choice.

   For example, if you wanted to work in the `/tmp/testdmf` directory:

```
$ cp /usr/share/doc/dmf-*/info/sample/sample_sitelib.C /tmp/testdmf/sitelib.C
$ cp /usr/share/doc/dmf-*/info/sample/sample_sitelib.mk  /tmp/testdmf/sitelib.mk
```

2. In the makefile, specify the stem from which the library filename and source code filename will be derived by editing the value for the `SITELIB` parameter. For example, to use a stem of `sitelib` (that is, `sitelib.so` for the library and `sitelib.c` for the source code file):

   ```
   SITELIB=sitelib
   ```

   Note: Although you can set the `SITELIB` value to something other than `sitelib` for testing purposes, when you actually want to run with DMF, it must be `sitelib`.

3. Read the comments at the start of each subroutine and alter the supplied code to suit your requirements. As supplied, each subroutine is disabled. To enable one or more subroutines, modify the `SiteFncMap` variable at the bottom of the source file (in our example, `sitelib.C`).

   Note: The name of the `SiteFncMap` variable is fixed and cannot be altered. However, you can change the names of the site-defined subroutines such as `SiteCreateContext()`.

4. Build the `sitelib.so` library by using the `make`(1) command:

   ```
   $ make -f sitelib.mk
   ```

5. Print a list of the subroutines that have been enabled and visually verify that it is what you expect:

   ```
   $ make -f sitelib.mk verbose
   ```

6. Install the library on a DMF server, which requires you to be the root user:

```
$ su
# make -f sitelib.mk install
```

Note: You do not need to install sitelib.so on a machine that functions only as a DMF client.

For subroutines that affect the operation of the DMF daemon, library server, or MSP, you must wait for a minute or so for the new sitelib.so library to be noticed. You will see a message in the relevant log file when this happens.

7. Test your new library by monitoring the relevant log file with tail -f while you present test cases to DMF. You may also find it useful to have a Resource Watcher configured and running or to use dmstat.

Considerations for Writing Custom Subroutines

As you write your own custom subroutines, be aware of the following:

- The sitelib.so file must be owned by root and must not be writable by anyone else, for security reasons. If these conditions are not met, DMF will ignore sitelib.so and use the default behavior.
- The sitelib.so library should not use the stdin, stdout, or stderr files as this could cause problems for DMF, possibly endangering data. For information about sending messages to users or to log files, see "DmaSendLogFmtMessage()" on page 763 and "DmaSendUserFmtMessage()" on page 764.
- If you overwrite the sitelib.so file while it is in use (for example by copying a new version of your file over the top of the old one), DMF processes may abort or run improperly. The DMF daemon may or may not be able to restart them properly.

  To update the file, you should do one of the following:

  – Use the mv(1) command to move the new file over the top of the old one, so that any existing DMF processes will continue to use the previous version of the file, which is now unlinked pending removal. The install target in the supplied makefile is also a safe way to update the file.

  - Delete the old file with rm(1) before installing the new one using cp, mv, or make install.
  - Shut down DMF while the update takes place.

  This warning also applies to changes to the DMF configuration file.

- Site-defined policy subroutines should not call subroutines in libdmfusr.so, such as DmuSettagByPathSync(). They are free to call member functions of classes defined in libdmfcom.H, such as DmuVolGroups_t::numVolGroups().
- At times, the site-defined subroutines may be called many times in rapid succession. They should therefore be as efficient as possible, avoiding any unnecessary processing, especially of system calls.

  For example, when dmfsfree is invoked to prevent a filesystem from filling, site-defined subroutines may be called one or more times for every file in the filesystem as dmfsfree prepares its list of candidates prior to migrating and/or freeing some of them. If the functions are slow, DMF may not be able to react to the situation in time to prevent the filesystem from filling.

sitelib.so Data Types

The data types described in this section are defined in libdmfadm.H. The information in this section is provided as a general description and overall usage outline. Other data types that are referenced in this file are defined in libdmfcom.H; see Appendix B, "DMF User Library libdmfusr.so" on page 681.

Note: For the most current definitions of these types, see the libdmfadm.H file.

DmaContext_t

The DmaContext_t object stores information for DMF in order to provide continuity from one subroutine call to the next. It is an opaque object that is created when a DMF process first loads sitelib.so and it exists until that process unloads it. This context is provided as a parameter for each of the site-defined policy subroutines.

Site-defined subroutines cannot directly access the information held in the context, but they can obtain information from it by using the following subroutines:

- "DmaGetContextFlags()" on page 759
- "DmaGetProgramIdentity()" on page 762
- "DmaGetUserIdentity()" on page 762

Site-defined subroutines can also store their own information in the context and retrieve it on subsequent calls by using the following subroutines:

- "DmaSetCookie()" on page 765
- "DmaGetCookie()" on page 759

DmaFrom_t

The DmaFrom_t object specifies the type of policy statement being evaluated.

There are the following possible values:

DmaFromAgeWeight	Indicates that an AGE_WEIGHT policy statement is being evaluated.
DmaFromSpaceWeight	Indicates that a SPACE_WEIGHT policy statement is being evaluated.
DmaFromVgSelect	Indicates that a SELECT_MSP or SELECT_VG policy statement is being evaluated.

DmaIdentity_t

The DmaIdentity_t object provides information, if known, about the program calling the site-defined subroutine and the user whose request generated the call.

The public member fields and functions of this class are as follows:

`realm_type`

Specifies the environment in which the type of data that is contained in the `realm_data` field is meaningful.

The following settings are defined:

- `DMF_REALM_UNIX` means that the `unix_1` member of `realm_data` contains valid information
- `DMF_REALM_UNKNOWN` means that `realm_data` is not reliable

`realm_data`

Specifies user identity information that is specific to the environment defined by `realm_type`. Only the `unix_1` member of the union is defined for the `realm_type` of `DMF_REALM_UNIX`.

If the UID and/or GID values are `0xffffffff`, the values are not reliable.

`logical_name`

Specifies a character string containing the program name of the process. This may be an absolute or relative pathname. If the value is `unknown`, the program name was unavailable.

`product_name_and_revision`

Specifies a character string containing the product name and revision (for example, `DMF_3.1.0.0`).

`locale_1`

Specifies a character string containing the locale value. See the `locale`(1) man page.

`host`

Specifies a character string containing the host on which the `DmaIdentity_t` originated.

`pid`

Specifies the process ID where the `DmaIdentity_t` originated.

instance_id

Specifies a further refinement of the PID field. Because a process may create more than one DmaIdentity_t, this value is incremented by one for each new DmaIdentity_t.

os_type

Specifies a character string containing a description of the operating system where the DmaIdentity_t originated.

os_version

Specifies a character string containing a description of the operating system version where the DmaIdentity_t originated.

cpu_type

Specifies a character string containing a description of the CPU type where the DmaIdentity_t originated.

Note: Any of the descriptive character strings may be set to unknown if the field's true value cannot be determined.

DmaLogLevel_t

The DmaLogLevel_t object specifies the level of a message. The administrator may select a log level in the DMF configuration file; messages with a less severe level than what is specified in the configuration file will not appear in the log.

DmaRealm_t

The DmaRealm_t object specifies the realm. Only the UNIX realm is supported.

DmaRecallType_t

The DmaRecallType_t object specifies the type of kernel recall being performed.

SiteFncMap_t

The SiteFncMap_t object specifies the site subroutine map. The various DMF processes that can call subroutines in sitelib.so look for a variable named SiteFncMap, of type SiteFncMap_t, in the sitelib.so library. It then uses the addresses provided in this variable to find the site-defined subroutines. If the variable is not found, DMF will not make any calls to subroutines in sitelib.so.

Site-Defined Policy Subroutines

DMF looks for the variable named SiteFncMap, of type SiteFncMap_t, in the sitelib.so library. It then uses the addresses provided in this variable to find site-defined subroutines listed in this section. You can provide any number of these subroutines in the sitelib.so library.

SiteArchiveFile()

The SiteArchiveFile() subroutine allows sites some control over the DMF archive requests. It is invoked when a dmarchive(1) command is issued to copy data directly to secondary storage or when one of the following libdmfusr.so subroutines is called:

```
DmuArchiveAsync()
DmuArchiveSync()
```

This subroutine is not called when automated space management migrates a file.

Caution: If SiteArchiveFile() is implemented, it takes precedence over any when clause being used to control MSP, volume group (VG), or migrate group (MG) selection, whether or not SiteWhen() has been implemented.

If this subroutine returns a value other than DmuNoError, the archive request will be rejected. The subroutine may not issue log messages, but it can issue messages to the user.

The prototype is as follows:

```
typedef  DmuError_t (*SiteArchiveFile_f) (
        const   DmaContext_t      dmacontext,
        const   DmuFullstat_2_t   *fstat,
```

```
const   char              *src_path,
const   char              *dst_path,
const   DmuFhandle_t      *dst_fhandle,
const   int               flags,
const   DmuVolGroups_t    *policy_volgrps,

const   DmuPriority_t     user_priority,
const   int               user_flags,
const   DmuVolGroups_t    *user_volgrps,

        DmuPriority_t     *operative_priority,
        int               *operative_flags,
        DmuVolGroups_t    *operative_volgrps);
```

The parameters are as follows:

dmacontext	Refers to the context established when sitelib.so was loaded.
fstat	Specifies the DmuFullstat_2_t information of the target file for the archive request.
src_path	Specifies the pathname of the source file for the archive request.
dst_path	Specifies the pathname of the destination file for the archive request.
dst_fhandle	Specifies the DmuFhandle_t of the destination file for the archive request.
flags	Specifies whether the SiteArchiveFile() subroutine is called for the first time (0) or is replayed (nonzero). SiteArchiveFile() can be called multiple times for the same request. For example, if dmfdaemon is not running, a dmarchive request will periodically try to establish a connection with it, and SiteArchiveFile() may be called. If flags is 0, this is the first time that SiteArchiveFile() has been called for a particular request. When a request is replayed, DMF reevaluates the parameters to SiteArchiveFile() before calling it.

`policy_volgrps`

Specifies an input parameter that contains the MSPs, VGs, and MGs that have been selected by the policy statements in the DMF configuration file.

`user_priority`
`user_flags`
`user_volgrps`

Contains information entered by the user as a `dmarchive` parameter (where supported) or as a parameter to one of the following `libdmfusr.so` subroutines:

`DmuArchiveAsync()`
`DmuArchiveSync()`

`operative_priority`
`operative_flags`
`operative_volgrps`

Contains the information that will be used when the request is made to `dmfdaemon`. These are all both input and output parameters. You can alter the `operative_flags` and `operative_volgrps` values. (Currently, `operative_priority` is ignored. For compatibility with future releases of DMF, it is recommended that you do not alter the value of this parameter.) If you alter `operative_volgrps`, take care that it expands to a non-overlapping set of MSPs and VGs when all the group members of the MGs are considered.

`SiteCreateContext()`

The `SiteCreateContext()` subroutine provides the opportunity to create a site-specific setup. It is called when `sitelib.so` is loaded. If no such setup is required, it need not be implemented. If this subroutine returns anything other than `DmuNoError`, no other subroutines in `sitelib.so`, including `SiteDestroyContext()`, will be called by the current process, unless `sitelib.so` is changed and therefore reloaded.

This subroutine may not issue messages to the user because the user details are unknown at the time it is invoked. If it is invoked by a program with a log file, such

as dmfdaemon, it can issue log messages by calling DmaSendLogFmtMessage(). You can call DmaGetContextFlags() to determine if it can issue log messages.

The prototype is as follows:

```
typedef DmuError_t  (*SiteCreateContext_f)(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

dmacontext	Refers to the context established when sitelib.so was loaded.

SiteDestroyContext()

The SiteDestroyContext() subroutine provides the opportunity for site-specific cleanup. It is called when sitelib.so is unloaded. If no such cleanup is required, it need not be implemented. This subroutine may not issue messages to the user because the user details are no longer valid at the time it is invoked. If it is invoked by a program with a log file, such as dmfdaemon, it can issue log messages by calling DmaSendLogFmtMessage(). You can call DmaGetContextFlags() to determine if it can issue log messages.

The prototype is as follows:

```
typedef void  (*SiteDestroyContext_f)(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

dmacontext	Refers to the context established when sitelib.so was loaded.

SiteKernRecall()

The SiteKernRecall() subroutine allows sites some control over kernel requests to recall a file. It is invoked when DMF receives a kernel request to recall a file. For example, a read() system call for a file that is currently in OFL state would result in SiteKernRecall() being called. The dmget command or the equivalent libdmfusr.so library call would not result in a call to SiteKernRecall().

This subroutine may accept or reject the request or change its priority; no other changes are possible. If the subroutine returns a value other than `DmuNoError`, the request will be rejected.

Note: `offset` and `length` pertain to the range of the file that the user's I/O request referenced, not the byte range that `dmfdaemon` will actually recall.

The subroutine may not issue messages to the user, but it can issue messages to the DMF daemon log.

The prototype is as follows:

```
typedef  DmuError_t (*SiteKernRecall_f) (
                DmaContext_t    dmacontext,
        const   DmuFullstat_2_t *fullstat,
        const   DmuFhandle_t    *fhandle,
                uint64_t        offset,
                uint64_t        length,
                DmaRecallType_t recall_type,
                DmuPriority_t   *operative_priority);
```

The parameters are as follows:

`dmacontext`	Refers to the context established when `sitelib.so` was loaded.
`fullstat`	Specifies the `DmuFullstat_2_t` of the file being recalled.
`fhandle`	Specifies the `DmuFhandle_t` of the file being recalled.
`offset`	Pertains to the range of the file that the user's I/O request referenced.
`length`	Pertains to the length of the file that the user's I/O request referenced.
`recall_type`	Specifies the type of recall.
`operative_priority`	Specifies the priority of the operation. `operative_priority` is both an input and an output parameter and you can alter the value. Valid values are in the range `DMU_MIN_PRIORITY` to `DMU_MAX_PRIORITY`. Higher values imply higher

priority. Only volume groups use priority; disk, DCM, and FTP MSPs ignore it.

SitePutFile()

The SitePutFile() subroutine allows sites some control over the DMF put requests. It is invoked when a dmput command is issued or when one of the following libdmfusr.so subroutines is called:

DmuPutByPathAsync()
DmuPutByPathSync()
DmuPutByFhandleAsync()
DmuPutByFhandleSync()

This subroutine is not called when automated space management migrates a file.

Caution: If SitePutFile() is implemented, it takes precedence over any when clause being used to control MSP, VG, or MG selection, whether or not SiteWhen() has been implemented.

If this subroutine returns a value other than DmuNoError, the put request will be rejected. The subroutine may not issue log messages, but it can issue messages to the user.

The prototype is as follows:

```
typedef  DmuError_t (*SitePutFile_f) (
        const   DmaContext_t    dmacontext,
        const   DmuFullstat_2_t   *fstat,
        const   char            *path,
        const   DmuFhandle_t    *fhandle,
        const   int             flags,
        const   DmuVolGroups_t  *policy_volgrps,

        const   DmuPriority_t   user_priority,
        const   int             user_flags,
        const   DmuByteRanges_t *user_byteranges,
        const   DmuVolGroups_t  *user_volgrps,

                DmuPriority_t   *operative_priority,
```

```
                int             *operative_flags,
                DmuByteRanges_t *operative_byteranges,
                DmuVolGroups_t  *operative_volgrps);
```

The parameters are as follows:

`dmacontext`	Refers to the context established when `sitelib.so` was loaded.
`fstat`	Specifies the `DmuFullstat_2_t` information of the target file for the `put` request.
`path`	Specifies the pathname of the target file for the `put` request (if known) or `NULL`.
`fhandle`	Specifies the `DmuFhandle_t` of the target file for the `put` request.
`flags`	Specifies whether the `SitePutFile()` subroutine is called for the first time (`0`) or is replayed (nonzero). `SitePutFile()` can be called multiple times for the same request. For example, if `dmfdaemon` is not running, a `dmput` request will periodically try to establish a connection with it, and `SitePutFile()` may be called. If `flags` is `0`, this is the first time that `SitePutFile()` has been called for a particular request. When a request is replayed, DMF reevaluates the parameters to `SitePutFile()` before calling it.
`policy_volgrps`	Specifies an input parameter that contains the MSPs, VGs, and MGs that have been selected by the policy statements in the DMF configuration file.

`user_priority`
`user_flags`
`user_byteranges`
`user_volgrps`
	Contains information entered by the user as a `dmput` parameter (where supported) or as a parameter to one of the following `libdmfusr.so` subroutines:
	
	`DmuPutByPathAsync()`
	`DmuPutByPathSync()`
	`DmuPutByFhandleAsync()`
	`DmuPutByFhandleSync()`

operative_priority
operative_flags
operative_byteranges
operative_volgrps

Contains the information that will be used when the request is made to dmfdaemon. These are all both input and output parameters. You can alter the operative_flags, operative_byteranges, and operative_volgrps values. (Currently, operative_priority is ignored. For compatibility with future releases of DMF, it is recommended that you do not alter the value of this parameter.) If you alter operative_volgrps, take care that it expands to a non-overlapping set of MSPs and VGs when all the group members of the MGs are considered.

SiteWhen()

The SiteWhen() subroutine provides the opportunity to supply the value for the sitefn variable in when clauses in the following parameters:

AGE_WEIGHT
SPACE_WEIGHT
SELECT_MSP
SELECT_VG

This subroutine and the sitefn variable in when clauses are not supported for the SELECT_LOWER_VG parameter.

Caution: If SitePutFile() or SiteArchiveFile() is implemented, it takes precedence over any when clause being used to control MSP, VG, or MG selection, whether or not SiteWhen() has been implemented.

For example,

```
SELECT_VG       tp9840  when uid = archive or sitefn = 6
```

If this subroutine is unavailable, either because it was not implemented or because the `sitelib.so` library is not accessible, the expression using `sitefn` is evaluated as being false. Therefore, the example above would be treated as if it were the following:

```
SELECT_VG       tp9840  when uid = archive or false
```

Or:

```
SELECT_VG       tp9840  when uid = archive
```

If a policy stanza contains multiple references to `sitefn`, it is possible that the subroutine is only called once and the value returned by that call may be used for several substitutions of `sitefn`. Therefore, a policy that contains the following will not necessarily call the subroutine three times:

```
AGE_WEIGHT      -1     0        when sitefn < 10
AGE_WEIGHT      1      .1
SPACE_WEIGHT    1      1e-6     when sitefn != 11
SPACE_WEIGHT    2      1e-9     when sitefn > 19
SPACE_WEIGHT    3.14   1e-12
```

The subroutine can issue log messages in some circumstances and user messages in others. You can call `DmaGetContextFlags()` to determine what kind of messages are possible.

The prototype is as follows:

```
typedef int (*SiteWhen_f) (
        const DmaContext_t    dmacontext,
        const DmuFullstat_2_t *fstat,
        const DmuFhandle_t    *fhandle,
              DmaFrom_t       fromtyp);
```

The parameters are as follows:

dmacontext	Refers to the context established when sitelib.so was loaded.
fstat	Specifies the DmuFullstat_2_t of the file being evaluated.
fhandle	Specifies the DmuFhandle_t of the file being evaluated.
fromtyp	Indicates what kind of policy is being evaluated.

Helper Subroutines for sitelib.so

This section describes optional subroutines that may be called from sitelib.so and are present in the processes that load sitelib.so.

DmaConfigStanzaExists()

The DmaConfigStanzaExists() subroutine checks whether a specified stanza exists in the DMF configuration file.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
DmaBool_t
DmaConfigStanzaExists(
        const   DmaContext_t    dmacontext,
        const   char            *type,
        const   char            *stanza);
```

The parameters are as follows:

`dmacontext`	Specifies the `DmaContext_t` parameter passed as input to all site-defined policy subroutines, such as `SitePutFile()`.
`type`	Specifies the type of the stanza being checked.
`stanza`	Specifies the name of the stanza being checked.

For example, if the DMF configuration file contained the following:

```
define  /dmf1
        TYPE            filesystem
        POLICIES        space_policy vg_policy
enddef
```

Then the following call would return true:

```
DmaConfigStanzaExists(dmacontext, "filesystem","/dmf1");
```

DmaGetConfigBool()

The `DmaGetConfigBool()` subroutine extracts parameter values of type `DmaBool_t` from the specified stanza in the DMF configuration file. If there is no such parameter definition or if it exists but with a missing or improper value, then the default is used.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
DmaBool_t
DmaGetConfigBool(
        const   DmaContext_t    dmacontext,
        const   char            *stanza,
        const   char            *param,
                DmaBool_t       default_val);
```

The parameters are as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().
stanza	Specifies the name of the stanza being searched.
param	Specifies the name of the parameter for which DmaGetConfigBool() is searching.
default_val	Specifies the value to use if param is not found in stanza or if param has a missing or invalid value.

DmaGetConfigFloat()

The DmaGetConfigFloat() subroutine extracts parameter values of type float from the specified stanza in the DMF configuration file. If there is no such parameter definition or if it exists but with a missing or invalid value, the default is used.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
float
DmaGetConfigFloat(
        const   DmaContext_t    dmacontext,
        const   char            *stanza,
        const   char            *param,
                float           default_val,
                float           min,
                float           max);
```

The parameters are as follows:

`dmacontext`	Specifies the `DmaContext_t` parameter passed as input to all site-defined policy subroutines, such as `SitePutFile()`.
`stanza`	Specifies the name of the stanza being searched.
`param`	Specifies the name of the parameter for which `DmaGetConfigFloat()` is searching.
`default_val`	Specifies the value to use if `param` is not found in `stanza` or if `param` has a missing or invalid value.
`min`	Defines the minimum valid value.
`max`	Defines the maximum valid value.

DmaGetConfigInt()

The `DmaGetConfigInt()` subroutine extracts parameter values of type `int64_t` from the specified stanza in the DMF configuration file. If there is no such parameter definition or if it exists but with a missing or invalid value, then a default value is used.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
int64_t
DmaGetConfigInt(
        const   DmaContext_t    dmacontext,
        const   char            *stanza,
        const   char            *param,
                int64_t         default_val,
                int64_t         min,
                int64_t         max);
```

The parameters are as follows:

`dmacontext`	Specifies the `DmaContext_t` parameter passed as input to all site-defined policy subroutines, such as `SitePutFile()`.

stanza	Specifies the name of the stanza being searched.
param	Specifies the name of the parameter for which DmaGetConfigInt() is searching.
default_val	Specifies the value to use if param is not found in stanza or if param has a missing or invalid value.
min	Defines the minimum valid value.
max	Defines the maximum valid value.

DmaGetConfigList()

The DmaGetConfigList() subroutine returns a pointer to an array of words found in the parameter in the specified stanza. The items value points to a block of memory containing an array of string pointers are well as the strings themselves; the end of the array is marked by a NULL pointer. The block of memory has been allocated by the malloc() subroutine and can be released with the free() subroutine if desired. The caller is responsible for releasing this memory.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
DmaBool_t
DmaGetConfigList(
        const   DmaContext_t    dmacontext,
        const   char            *stanza,
        const   char            *param,
                char            *** items);
```

The parameters are as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().
stanza	Specifies the name of the stanza being searched.
param	The name of the parameter for which DmaGetConfigList() is searching.

`items`	Specifies an output value that points to a block of memory containing an array of string pointers as well as the strings themselves; the end of the array is marked by a `NULL` pointer.

DmaGetConfigStanza()

The `DmaGetConfigStanza()` subroutine return a pointer to an array of parameters and values for the specified stanza in the DMF configuration file. (That is, it provides the entire stanza, after comments have been removed.) The `items` value points to a block of memory containing an array of structures with string pointers as well as the strings themselves; the end of the array is marked by a `NULL` pointer. The block of memory has been allocated by the `malloc()` subroutine and can be released with the `free()` subroutine if desired. The caller is responsible for releasing this memory.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
DmaBool_t
DmaGetConfigStanza(
        const   DmaContext_t    dmacontext,
        const   char            *stanza,
                DmaConfigData_t **items);

}
```

The parameters are as follows:

`dmacontext`	Specifies the `DmaContext_t` parameter passed as input to all site-defined policy subroutines, such as `SitePutFile()`.
`stanza`	Specifies the name of the stanza being searched.
`items`	Specifies an output value that points to a block of memory containing an array of structures with string

pointers as well as the strings themselves; the end of the array is marked by a NULL pointer.

DmaGetConfigString()

Extracts a string from the specified stanza in the DMF configuration file and returns it. If there is no such parameter definition, the default is used. If the parameter exists but with a missing value, the null string (which is a valid value) is returned.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
void
DmaGetConfigString(
        const   DmaContext_t    dmacontext,
        const   char            *stanza,
        const   char            *param,
        const   char            *default_val,
        DmuStringImage_t        &result);
```

The parameters are as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().
stanza	Specifies the name of the stanza being searched.
param	Specifies the name of the parameter for which DmaGetConfigString() is searching.
default_val	Specifies the value to use if param is not found in stanza. If param is found in stanza but has a missing value, the null string is returned.
result	Specifies an output parameter, containing the result.

DmaGetContextFlags()

The DmaGetContextFlags() subroutine determines if a given subroutine can issue log messages or issue user messages.

Note: If DmaFlagContextValid() is not set in the return value, no use should be made of any other bits.

DmaGetContextFlags() can return the following values, which may be OR'd together:

DmaFlagContextValid	Indicates that the context is valid.
DmaFlagLogAvail	Indicates that DmaSendLogFmtMessage may be called.
DmaFlagMsgAvail	Indicates that DmaSendUserFmtMessage may be called.

The prototype is as follows:

```
uint64_t
DmaGetContextFlags(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().

DmaGetCookie()

The DmaGetCookie() subroutine returns the cookie that was stored in dmacontext by a call to DmaSetCookie(). If a NULL value is returned, either the context is invalid or the cookie was not set.

The prototype is as follows:

```
void *
DmaGetCookie(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().

DmaGetDaemonMigGroups()

The DmaGetDaemonMigGroups subroutine returns the list of configured migrate groups.

The prototype is as follows:

```
const DmuVolGroups_t *
DmaGetDaemonMigGroups(
        const   DmaContext_t    dmacontext)
```

The parameter is as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().

DmaGetDaemonVolAndMigGroups()

The DmaGetDaemonVolAndMigGroups() subroutine returns the MSPs, VGs, and MGs that dmfdaemon is currently configured to use.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
const DmuVolGroups_t *
DmaGetDaemonVolAndMigGroups(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().

DmaGetDaemonVolGroups()

The DmaGetDaemonVolGroups() subroutine returns the MSPs and VGs that dmfdaemon is currently configured to use.

Note: Values in the configuration file may change while DMF is running.

The prototype is as follows:

```
const DmuVolGroups_t *
DmaGetDaemonVolGroups(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().

DmaGetMigGroupMembers()

The DmaGetMigGroupMembers subroutine returns group members of the given migrate group. The return value must be explicitly released by the caller using the free() subroutine.

```
char **
DmaGetMigGroupMembers(
        const   DmaContext_t    dmacontext,
        const   char     *mg_name)
```

The parameters are as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().

mg_name	The name of the migrate group

DmaGetProgramIdentity()

The DmaGetProgramIdentity() subroutine returns a pointer to the program DmaIdentity_t object in the dmacontext parameter.

Note: The program DmaIdentity_t object should not be confused with the user DmaIdentity_t object that is returned by "DmaGetUserIdentity()" on page 762. The user identity is usually of much more interest when applying site policies because it defines who is actually making the request as opposed to what process is negotiating the site policies.

The prototype is as follows:

```
const DmaIdentity_t *
DmaGetProgramIdentity(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().

DmaGetUserIdentity()

The DmaGetUserIdentity() subroutine returns a pointer to the user DmaIdentity_t object in the dmacontext parameter.

The user DmaIdentity_t object contains as much information as could be reliably gathered regarding the identity of the originator of the request. For example, the user identity in the SitePutFile() policy subroutine would identify the process (such as dmput) that made the original DmuPutByPathSync() libdmfusr call.

If DmaGetUserIdentity() is called from within SiteKernRecall(), it will return the identity of dmfdaemon. The identity of the user who initiated the read request that caused SiteKernRecall() to be called is unknown to DMF.

Within SiteCreateContext(), the user details may be as yet unknown; therefore, DmaGetUserIdentity() may return different values than if it is called with the

same context from another site-defined policy subroutine. In most cases, the user identity is determined after the call to `SiteCreateContext()`.

Under certain circumstances, some elements of the `DmaIdentity_t` structure may be unknown. For example, if a site-defined subroutine is called as a result of a command entered on a client machine running a release prior to DMF 3.1, some elements of the user identity may be unknown.

The prototype is as follows:

```
const DmaIdentity_t *
DmaGetUserIdentity(
        const   DmaContext_t    dmacontext);
```

The parameter is as follows:

`dmacontext` Specifies the `DmaContext_t` parameter passed as input to all site-defined policy subroutines, such as `SitePutFile()`.

`DmaSendLogFmtMessage()`

The `DmaSendLogFmtMessage()` subroutine formats and issues log messages, if log messages are possible. The messages will potentially appear in the calling program's log depending upon the `DmaLogLevel_t` of the message and the log level selected by the administrator in the DMF configuration file. If log messages are not possible, `DmaSendLogFmtMessage()` silently discards the message.

The prototype is as follows:

```
void
DmaSendLogFmtMessage(
        const   DmaContext_t    dmacontext,
                DmaLogLevel_t   log_level,
        const   char            *name,
        const   char            *format,
                ...);
```

The parameters are as follows:

`dmacontext` Specifies the `DmaContext_t` parameter passed as input to all site-defined policy subroutines, such as `SitePutFile()`.

log_level	Specifies the level of the message.
name	Specifies a string that is included as part of the log message.
format	Specifies the format for the message that will be printed in the log. It looks like a printf(3S) format. Do not include \n as part of the message. If you want to print more than one line to the log, make multiple calls to DmaSendLogFmtMessage().

For example, the following will issue an error message to the calling program's log:

```
DmaSendLogFmtMessage (dmacontext, DmaLogErr,
              "SiteCreateContext", "sitelib.so problem errno %d",
              errno);
```

DmaSendUserFmtMessage()

The DmaSendUserFmtMessage() subroutine formats and sends messages to the user, if user messages are possible. The messages will potentially appear as output from commands such as dmput and dmget, depending upon the severity of the message and the level of message verbosity selected by the user. If user messages are not possible, DmaSendUserFmtMessage() silently discards the message.

The prototype is as follows:

```
void
DmaSendUserFmtMessage(
        const   DmaContext_t    dmacontext,
                DmuSeverity_t   severity,
        const   char            *position,
                int             err_no,
        const   char            *format,
                ...);
```

The parameters are as follows:

dmacontext	Specifies the DmaContext_t parameter passed as input to all site-defined policy subroutines, such as SitePutFile().
severity	Specifies the severity of the message.

position	Specifies a string that can be included in the message. This string may be set to NULL.
err_no	Specifies that if err_no is non-zero, the results of strerror(err_no) will be included in the message.
format	Specifies the format for the message that will be sent to the user. It looks like a printf(3S) format. It is not necessary to put \n at the end of the message.

DmaSetCookie()

The DmaSetCookie() subroutine stores a pointer to site-defined subroutine information in dmacontext. This pointer may be retrieved by a call to DmaGetCookie(). The site-defined subroutines are responsible for memory management of the space pointed to by the cookie parameter.

The prototype is as follows:

```
void
DmaSetCookie(

        const   DmaContext_t    dmacontext,
                void            *cookie);
```

The parameters are as follows:

dmacontext	Specifies the DmaContext_t parameter that is passed as input to all site-defined policy subroutines, such as SitePutFile().
cookie	Specifies a pointer to information that sitelib.so subroutines want to retain while the dmacontext is valid.

Third-Party Backup Package Configuration

This appendix discusses the following:

- "Using Third-Party Backup Packages" on page 767
- "ASG-Time Navigator" on page 769
- "`taskgroup` Object Example for Third-Party Backups" on page 770

Using Third-Party Backup Packages

This section discusses the following:

- "Overview of Using Third-Party Backup Packages" on page 767
- "`do_predump.sh` Script" on page 768
- "`DUMP_DATABASE_COPY` Configuration File Parameter in the `taskgroup` Object" on page 768
- "Hard Deletes and Third-Party Backup" on page 769
- "Unprivileged Users" on page 769
- "Procedure to Use a Third-Party Backup Package" on page 769

Overview of Using Third-Party Backup Packages

Many backup and recovery software packages make backup copies of files by opening and reading them using the standard UNIX or Linux system calls. In a filesystem managed by DMF, this causes files that are offline to be recalled back to disk before they can be backed up. If you have a managed filesystem in which a high percentage of the files are offline, you may see a large amount of media or other activity caused by the backup package when it initially does its backups. You should take this behavior into account when deciding whether or not to use such backup packages with filesystems managed by DMF.

Some third-party backup packages can use a DMF library to perform backups in a manner that is aware of DMF. When the feature that is aware of DMF is enabled, these packages will not cause offline (`OFL`) files to be recalled during a backup.

Dual-state (DUL) files will be backed up as if they were offline, which will reduce the time and space needed for a backup.

do_predump.sh Script

DMF provides a script called do_predump.sh that is meant to be run just prior to a backup of the DMF filesystems using a third-party backup package. The do_predump.sh script does the following:

- *(Optional)* Migrates all eligible files to dual-state
- *(Optional on a DCM MSP system)* Copies all eligible DCM MSP files to dual-residency state
- *(Optional)* Performs a snapshot of the databases by using dmsnap

To use do_predump.sh, do the following:

1. Configure the backup package to run do_predump.sh as the pre-backup command. For details, see the application-specific information in Appendix D, "Third-Party Backup Package Configuration" on page 767.
2. Define a task group in the dmf.conf file that is referred to by the dmdaemon object. In the supplied configurations, this task group is called dump_tasks.

The parameters do_predump.sh uses are as follows:

- DUMP_DATABASE_COPY
- DUMP_FLUSH_DCM_FIRST
- DUMP_FILE_SYSTEMS
- DUMP_MIGRATE_FIRST

For more information, see "taskgroup Object Parameters" on page 312.

DUMP_DATABASE_COPY Configuration File Parameter in the taskgroup Object

The taskgroup object also supports the DUMP_DATABASE_COPY for third-party backup only. It specifies the path to a directory where a snapshot of the DMF databases will be placed when do_predump.sh is run. The third-party backup application should be configured to back up this directory. By default, a snapshot will not be taken.

Hard Deletes and Third-Party Backup

Because hard-deletions normally use the same expiry time as backups, `run_hard_deletes.sh` is normally run from the same task group. The `DUMP_RETENTION` parameter should match the retention policy of the backup package. For an example stanza, see Example D-1 on page 770.

Unprivileged Users

Backups and restores must be run from the DMF server.

Only `root` can perform backups and restores. Although some third-party backup packages normally allow unprivileged users to restore their own files, unprivileged users cannot restore their own files from a DMF filesystem because doing so requires `root` privilege to set the DMF attribute.

Files backed up from a DMF filesystem should only be restored to a DMF filesystem. Otherwise, files that are offline (or treated as such) will not be recallable.

Procedure to Use a Third-Party Backup Package

To use a third-party backup package to back up DMF filesystems, do the following:

1. Configure the backup package to include the DMF filesystems in the backups.
2. Enable the feature that is aware of DMF on those filesystems.

ASG-Time Navigator

Note: SGI Support does not cover ASG-Time Navigator. For more information about ASG-Time Navigator, see www.asg.com and the ASG-Time Navigator manuals.

ASG-Time Navigator is high-performance backup and recovery software designed with intuitive graphical user interfaces (GUIs) to manage data in heterogeneous environments.

ASG-Time Navigator is aware of DMF and supports a broad range of servers and client operating systems. It also supports a wide range of SAN hardware and tape

libraries. ASG-Time Navigator by default uses a proprietary protocol for all data transfers.

To make ASG-Time Navigator aware of a managed filesystem, add lines resembling the following to the *full-TimeNavigator-installation-path*/Conf/parameters file, where */dmfusr1* and */dmfusr2* are the managed filesystems:

```
<parameter name="bapi_fs">
<value>/dmfusr1</value>
<value>/dmfusr2</value>
</parameter>
```

Using the ASG-Time Navigator GUI, you can define **backup classes** to select which directories you want to back up. You can also vary the granularity for backup and restore, such as file, directory, or class level.

To use DMF's do_predump.sh script with ASG-Time Navigator, set up ASG-Time Navigator to use a precommand as follows:

- In the **Advanced** tab of the backup strategy, specify the following as the preprocessing command:

  /usr/lib/dmf/do_predump.sh *daemon dump_tasks*

  where:

  daemon Name of the dmdaemon object in the DMF configuration file

  dump_tasks Name of the task group specifying the parameters related to backups

- Ensure that DMF's DUMP_RETENTION parameter matches the retention value of the cartridge pool associated with backing up the DMF filesystem.

taskgroup Object Example for Third-Party Backups

Example D-1 taskgroup Object for Third-Party Backups

```
define dump_tasks
        TYPE                    taskgroup
        RUN_TASK                $ADMINDIR/run_hard_deletes.sh at 23:00
        DUMP_RETENTION          4w

        DUMP_MIGRATE_FIRST      yes
        DUMP_FLUSH_DCM_FIRST    yes
```

```
        DUMP_DATABASE_COPY      /path/to/db_snapshot
enddef
```

The above example is similar to Example 7-10, page 333, and Example 7-11, page 335, but has the following differences:

- The backups are not managed via the dump scripts
- There is a DCM MSP, so the `dmmigrate` command should be run before the backups are done
- A snapshot of the DMF databases will be placed in `/path/to/db_snapshot` when `do_predump.sh` is run

Converting from IRIX DMF to Linux® DMF

Note: This procedure must take place during a planned outage of the systems and filesystems managed by DMF. It is assumed that sites converting DMF from IRIX to Linux will obtain the help of SGI customer support. The following documentation is offered to familiarize you with the necessary steps.

This appendix describes the necessary steps to convert an IRIX DMF system to a Linux DMF system and provides an example using a single library server (LS).

You cannot copy DMF databases from an IRIX system to a Linux system because of binary incompatibility. Instead, you must dump the IRIX DMF databases to text on the IRIX system and load the resulting text file into the databases on the Linux system. However, you can move managed filesystems (that is, filesystems containing user files that DMF has migrated) from an IRIX system to a Linux system.

Procedure E-1 Converting from IRIX DMF to Linux DMF

1. Discontinue all user activity for the duration of the IRIX to Linux conversion process.
2. If you have a tape MSP, you must convert it to a volume group (VG) in an LS **while still on IRIX** using `dmmsptols`.

   Note: The tape MSP is not available in the Linux DMF release.

   For more information, see the DMF 3.0 version of the *DMF Administrator's Guide for SGI InfiniteStorage* (007-3681-008).
3. Prepare the DMF databases on the IRIX system:
   a. Change the filesystem migration levels in the `dmf.conf` file to `none`.
   b. Run `dmdidle` and wait for activity to cease.
   c. Use `dmsnap` to back up the DMF databases.

      Alternatively, if time or disk space considerations are critical, it is acceptable to use the snapshot of the DMF databases that is generated in the

dmaudit_working_dir as part of step 4 below as the database backup copy, allowing you to skip this `dmsnap` step.

4. Audit the DMF databases to ensure that they are valid:

Caution: Do not proceed until you have obtained clean results for each step in turn.

   a. Run `dmaudit snapshot` and resolve all errors before moving on to step 4.b.

   b. Run `dmatvfy` *dmaudit_working_dir* and resolve all errors before moving on to step 4.c.

   c. Run `dmdskvfy` against all DCM and disk MSPs and resolve all errors before moving on to step 5.

   For more information, see the `dmaudit`(8) man page and *DMF Filesystem Audit Guide for SGI InfiniteStorage.*

5. Stop DMF on the IRIX system.

Caution: If DMF is started again on the IRIX system during or after this procedure, the databases captured during step 7 might not reflect reality, and loss of data might result if you use them.

6. Use `dmdbcheck` to verify the consistency of the DMF databases.

7. Dump all of the DMF databases to text from the snapshot taken in step 3c above. This should include the daemon database and the CAT and VOL tables for each LS database. For more information, see the `dmdump`(8) man page.

8. Sort the daemon and CAT text database records for better overall performance of the text-record load process. (The time to sort and load will be less than the time to load unsorted text records when the number of records is in the millions.) Do the following:

   - To sort the daemon text record file, use a command similar to the following, where *tmpdir* is a directory in a filesystem with sufficient free space for `sort` to complete the sort:

     ```
     # /bin/sort -t"|" -y -T tmpdir -k 1,1 -o sorted_daemontext daemontext
     ```

- To sort the CAT text record file, use a command similar to the following, where *tmpdir* is a directory in a filesystem with sufficient free space for `sort` to complete the sort:

  # **`/bin/sort -t"|" -y -T` *`tmpdir`* `-k 2,2 -o sorted_cattext cattext`**

For more information, see the `sort`(1) man page.

9. Set up the `/etc/dmf/dmf.conf` file on the Linux system. The conversion will be simpler if you name all of the FTP MSPs, disk MSPs, tape VGs, and LSs with the same names used on IRIX. This assumes that you do not already have MSPs, LSs, or VGs with these names on your Linux system.

   If you do change the name of an MSP or VG, you must convert the daemon database. For more information on how to perform this conversion, see the documentation in the `dmconvertdaemon` script.

10. Use `dmcheck` to ensure that your new `/etc/dmf/dmf.conf` file is valid on the Linux system.

11. Copy the text versions of the databases (which you created in step 7 and sorted in step 8) to the Linux system.

12. Load the database files from the text files on the Linux system. Use the following commands:

    Note: If you are loading the text records into an empty database, use the `-j` option on the `dmdadm`(8) and `dmcatadm`(8) commands to eliminate the unnecessary overhead of database journal records. If you are loading the records into a nonempty database, SGI recommends that you make a copy of the database before running the `dmdadm` and `dmcatadm` commands and that you do not use `-j` option.

    - `dmdadm` to load the daemon database file
    - `dmcatadm` to load the CAT records for each of the LS databases
    - `dmvoladm` to load the VOL records for each of the LS databases

13. Use `dmdbcheck` to check the consistency of databases on the Linux system.

14. Move all of the managed filesystems and DCM MSP filesystems from the IRIX system to the Linux system:

- If reusing the existing disks and the IRIX filesystem blocksize is supported by Linux (512, 1024, 2048, 4096, 8192, or 16384), you can simply move the disks from the IRIX system to the Linux system.
- If there is a disk resource upgrade or if the IRIX block size greater than what is supported in Linux, there will be new filesystems built under Linux. The old data must then be restored to these new filesystems. For information, see "Restoring Files from Backups" on page 643.

15. Start DMF on the Linux system.
16. Run `dmaudit` to verify the filesystems.

Example E-1 IRIX to Linux Conversion (Single LS)

In the following example, the IRIX system has a single LS named `ls1`. The example assumes that the `/tmp/dmfdatabases` directory has been created, is initially empty, and contains enough space to accommodate the text versions of the databases. The example also assumes that the `HOME_DIR` configuration parameter is set to `/dmf/home` on both systems. After completing steps 2 through 6 of Procedure E-1 on page 773, the daemon database and the CAT and VOL tables of the LS database are dumped to text, as follows:

```
$ dmdump -c /dmf/home/daemon > /tmp/dmfdatabases/daemon_txt
$ dmdump /dmf/home/ls1/tpcrdm.dat > /tmp/dmfdatabases/ls1_cat_txt
$ dmdump /dmf/home/ls1/tpvrdm.dat > /tmp/dmfdatabases/ls1_vol_txt
```

Next, the files in `/tmp/dmfdatabases` on the IRIX system are copied to `/tmp/dmftxtdb` on the Linux system. After creating the DMF configuration file on the Linux system, the databases are loaded on the Linux system, as follows:

```
$ dmdadm -u -c "load /tmp/dmftxtdb/daemon_txt"
$ dmcatadm -m ls1 -u -c "load /tmp/dmftxtdb/ls1_cat_txt"
$ dmvoldadm -m ls1 -u -c "load /tmp/dmftxtdb/ls1_vol_txt"
```

Now `dmdbcheck` is run to verify the consistency of the databases, as follows:

```
$ cd /dmf/home/daemon; dmdbcheck -a dmd_db
$ cd /dmf/home/ls1; dmdbcheck -a libsrv_db
```

Converting from TMF to OpenVault

The following provides an overview of the steps to convert from TMF to OpenVault. For assistance, contact SGI Professional Services.

Do the following:

1. Stop DMF.
2. Stop TMF.
3. Configure OpenVault. Ensure that the OpenVault library control program (LCP) uses the same ACSLS ID as TMF. For more information see:
   - "OpenVault Configuration Tasks" on page 491
   - *OpenVault Administrator Guide for SGI InfiniteStorage*
   - `ov_admin`(8) man page
4. Import the media into OpenVault using the `dmov_makecarts`(8) command, which adds the appropriate information so that tapes can be mounted via OpenVault and assigns the tapes to the OpenVault `dmf` application.

   For each cartridge type at your site, enter one of the following commands

   # **`dmov_makecarts -g`** ***cartgroup*** **`-t`** ***carttype lsname***

   # **`dmov_makecarts -g`** ***cartgroup*** **`-t`** ***carttype*** **`-v`** ***vg1,vg2 lsname***

   # **`dmov_makecarts -g`** ***cartgroup*** **`-t`** ***carttype taskgroup***

   where:

cartgroup	Specifies the cartridge group
carttype	Specifies the cartridge type
lsname	Specifies the library server
taskgroupname	Specifies the name of the `taskgroup` object

vg1,vg2	Specifies the volume groups

Note: In some situations, such as STK T100000A and T100000B tapes, the above procedure does not apply because you must use the `ov_import`(8) command with the `-b` option to set the `PartitionBitFormat` field when you load the tapes. For more information, contact SGI Professional Services.

5. Edit the DMF configuration file to set the `MOUNT_SERVICE` parameter to `openvault` (or delete the parameter). See:

   - "`device` Object Parameters" on page 348
   - "`drivegroup` Object Parameters" on page 397

Considerations for Partial-State Files

This section discusses the following:

- "Performance Cost Due to Lack of Linux Kernel Support" on page 779
- "Inability to Fulfill Exact Byte Range Requests" on page 780

Performance Cost Due to Lack of Linux Kernel Support

The Linux kernel does not provide underlying support for partial-state files. A partial-state file looks exactly like an offline file to the filesystem, and so all read requests for a partial-state file generate a DMF daemon read event, whether the byte range being read is actually already online or not. The DMF daemon will write an attribute to a partial-state file that includes the number and boundaries of each region so that any read event whose byte range is completely contained in an online region will return immediately to the kernel with no intervening recall. A read event whose byte range is not completely contained in an online region will result in the entire file being recalled.

Because there is no underlying support in the Linux kernel, the DMF partial-state file feature has a performance cost. The kernel cannot detect when a read request could be satisfied without a read event being generated to the DMF daemon, resulting in pseudo read events that cannot be absorbed by the system and therefore impact the system's performance. A performance degradation will be noticed if thousands of pseudo read events are being generated in a short period of time.

For example, if a very large file has a very large online region followed by a very small offline region and a process is doing a sequential read through the file using a small buffer size, each of the reads for the online region will result in a pseudo read event until finally a read for the offline region will cause the rest of the file to be brought back online. A single process doing this kind of operation might not impact the system, but tens or hundreds of simultaneous similar processes may. In this situation, it might be better to manually recall the file before doing the read.

Additionally, the pseudo read events will result in DMF daemon log-file entries for each read, and so the DMF `SPOOL_DIR` directory may experience a very significant increase in the amount of disk space that is consumed each day. If this is the case, the `SPOOL_DIR` directory will require maintenance (file removal) on a more frequent basis.

Inability to Fulfill Exact Byte Range Requests

User files can become partial-state either manually or automatically. The manual method involves using the byte-range parameters on the `dmput`(1) and `dmget`(1) commands. (See the man pages for a full description of the syntax of the byte-range specifications). You can use these commands to manually control which regions of a user file should be made online or offline, subject to the restrictions of the underlying filesystem and the maximum number of regions allowed in that filesystem.

All currently supported filesystems have a restriction that punching a hole in a file (to make a region offline) must take place on a fixed boundary size, usually on a 4096–byte block boundary. If a user requested an offline region from byte 10000 to byte 20000, the resulting offline byte range would be from byte 12288 to byte 16384. Offline regions are rounded inward, which might result in fewer bytes than specified being made offline, but will never result in more bytes than specified being made offline.

When requesting online regions, the byte addresses are rounded outward. So in the 10000-20000 byte address example, the resulting online region would be from byte 8192 to byte 20480 based on the idea that it is better to bring some extra bytes online than to not bring all of the bytes that were requested online.

It is entirely possible that a `dmput` or `dmget` request that specifies a byte-range parameter will result in no action on the file taking place. This is possible if the file is already in the requested state (just like using `dmget` on a `DUALSTATE` file before the introduction of partial-state files) or if the requested state would result in more than the maximum number or regions allowed by the filesystem per file. (See the `MAX_MANAGED_REGIONS` configuration file parameter in "`filesystem` Object" on page 350.) Because of the general inability of DMF to deliver the exact byte ranges requested, requests that do not deliver exact byte range results do not return an error. It is up to the caller to determine the exact state of the file after the request.

Case Study: Impact of Zone Size on Tape Performance

This appendix details an experiment with a 100 MB/s LTO4 drive, which is in the same performance class as the STK T10000A. The purpose of the test was to show the cost of having a small zone size (the `ZONE_SIZE` parameter, see "`volumegroup` Object" on page 410).

The `moverlog.`*yyyymmdd* log traces show two tests:

- In the first test, we migrated 200 512–MB files to tape using a `ZONE_SIZE` of `10g` (10 GB). This resulted in 10 zones.
- In the second test, we recalled all the files, changed the `ZONE_SIZE` to `499m` (499 MB), and remigrated the same 200 files. In the second test, each migrated file became its own zone (200 zones).

In the first test (with a `ZONE_SIZE` of `10g`), the tape drive achieved 118–MB/s per zone. This is the drive's full streaming rate. For example, the drive spent 89.6 seconds doing I/O to the first zone and only 1.48 seconds flushing the first zone:

```
12:49:54-V   102037-dmatwc process_completed_zone: Zone 1 written, chunks=21, bytes=10752000000
12:49:54-V   102037-dmatwc stats: idle=0.00, mount=32.27, skip=0.00, io=89.60, zone=1.48
12:49:54-V   102037-dmatwc stats: total chunks=21, mb=10752.000000, rate=118.05 mb/s
```

When the first migration test was complete, the `dmatwc` final statistics showed that the drive consistently achieved 114 MB/s, and the effective rate (if you include mount/unmount/zone/close/rewind time) was 89 MB/s (line breaks shown here for readability):

```
13:06:55-I   102037-dmatwc final_stats: idle=107.66, mount=32.27, skip=0.00, io=868.54,
   zone=20.84, close=81.29, unmount=34.19
13:06:55-I   102037-dmatwc final_stats: total sec = 1144.78, totalmb=101911.101562, rate=114.59 mb/s, effective
   rate=89.02 mb/s
```

In the second test (with a `ZONE_SIZE` of `499m`), the increased stop/start behavior of the drive meant that the drive only achieved about half of its native rate, or 67.28 MB/s (line breaks shown here for readability):

```
13:19:53-V   104013-dmatwc process_completed_zone: Req=4,6dc90 done, chunk=7, zone=4,
   chunklength=512000000, bytes=512000000
13:19:53-V   104013-dmatwc process_completed_zone: Zone 4 written, chunks=1, bytes=512000000
```

```
13:19:53-V   104013-dmatwc stats: idle=0.01, mount=31.88, skip=0.00, io=23.70, zone=5.93
13:19:53-V   104013-dmatwc stats: total chunks=1, mb=512.000000, rate=67.28 mb/s
```

When the second migration test was complete, the `dmatwc` final statistics show that the drive was only able to achieve 66 MB/s when it was doing I/O. Furthermore, 304.58 seconds were spent just flushing data (versus 20 seconds in the first test). Thus the effective rate in the second case was only 56 MB/s (line breaks shown here for readability):

```
13:48:57-I   104013-dmatwc final_stats: idle=114.54, mount=31.88, skip=0.00, io=1237.52,
   zone=304.58, close=82.74, unmount=34.09
13:48:57-I   104013-dmatwc final_stats: total sec = 1805.36, totalmb=102248.742188, rate=66.30 mb/s, effective
   rate=56.64 mb/s
```

Had we done a larger test and written an entire tape in each case, the mount, unmount, and close (rewind) time would have contributed much less to the effective bandwidth, and so the numbers would be even more dramatic.

You can obtain the statistics discussed in this appendix from the following log file:

SPOOL_DIR`/`*ls_name*`/moverlogs/`*hostname*`/moverlog.`*yyyymmdd*

For more information, see:

- "Improve Drive Performance with an Appropriate VG Zone Size" on page 105
- "LS Logs" on page 539

Historical Feature Information

This appendix contains the following:

- "End of Life for the Tape Autoloader API with DMF 2.6.3" on page 783
- "DMF Directory Structure Prior to DMF Release 2.8" on page 783
- "End of Life for the Tape MSP after DMF 3.0" on page 784
- "DMF User Library (`libdmfusr.so`) Update in DMF 3.1" on page 784
- "Downgrading and the Site-Tag Feature Introduced in DMF 3.1" on page 785
- "Downgrading and the Partial-State File Feature Introduced in DMF 3.2" on page 786
- "`dmaudit`(8) Changes in DMF 3.2" on page 787
- "Logfile Changes in DMF 3.2" on page 787
- "Possible DMF Database Lock Manager Incompatibility On Upgrades as of DMF 3.8.3" on page 788

End of Life for the Tape Autoloader API with DMF 2.6.3

With the release of DMF 2.6.3, DMF dropped support for the tape autoloader API. DMF supports OpenVault and TMF as tape mounting services. If you have not yet acquired OpenVault or TMF, do not upgrade to any version of DMF 2.6.3 or later.

DMF Directory Structure Prior to DMF Release 2.8

Beginning with DMF 2.8, DMF no longer supports multiple installed versions of DMF that can be made active via the `dmmaint`(8) program. While it is not necessary to delete any existing pre-2.8 versions of DMF, they will not accessible by the DMF 2.8 or later software and they can be removed at the convenience of the administrator.

The reason for this change is that the pre-2.8 DMF directory hierarchy of `/usr/dmf/dmbase` is no longer the target installation directory of DMF. Rather, DMF 2.8 and later binaries, libraries, header files, and man pages are installed directly into

the proper system locations and they are accessed directly from those locations without the use of symbolic file links.

When DMF 2.8 or later is installed, if the symbolic file link `/etc/dmf/dmbase` exists, it will be deleted. This link was used in pre-2.8 versions of DMF to access the active version of DMF, and as such, it was part of the administrators' initialization procedure to add this link to their `PATH` environment variable. Because it is no longer used in DMF 2.8 and later versions, it could cause an incorrect copy of a DMF command to be executed if an administrator's path included the link to be searched before the normal system binary locations. This way, even if the administrator neglects to remove the link from the path, it should not make any difference.

End of Life for the Tape MSP after DMF 3.0

DMF 3.0 was the last major release cycle that contained support for the tape MSP. The `dmatmsp` command is not included as part of any DMF 3.5 or later package. When the library server (LS) was introduced in DMF 2.7, the intention was for all existing tape MSPs to be converted to LSs eventually.

It is mandatory that you complete the conversion from tape MSPs to LSs before installing DMF 4.0 or later. SGI highly recommends that you install DMF 3.0.1 for the purpose of doing the conversion because the `dmmsptols` command in that release is much more efficient in terms of time and disk space than in any earlier release.

For more information regarding converting tape MSPs to LSs, see Chapter 14, "Library Servers and Media-Specific Processes" on page 531 or contact SGI Support.

DMF User Library (`libdmfusr.so`) Update in DMF 3.1

The DMF user library (`libdmfusr.so`) was modified significantly in DMF 3.1 and is not backwards compatible with applications written and linked with pre-3.1 versions of `libdmfusr.so`. The library's naming convention has also changed.

This change only impacts sites with site-written applications that link with `libdmfusr.so`. Any site that does have any such applications should immediately refer to Appendix B, "DMF User Library `libdmfusr.so`" on page 681 to find the steps required to keep your site applications operational.

Downgrading and the Site-Tag Feature Introduced in DMF 3.1

DMF 3.1 introduced the site tag feature; see dmtag(1). Site tags are stored in the DMF extended attribute on files. This means that if you have installed and run DMF 3.1 or later and wish to run an earlier version of DMF (pre-DMF 3.1), you must ensure that there are no nonzero site tags on files before installing the earlier version of DMF. Failure to do this will cause errors when running the earlier version of DMF.

Note: Restoring a file that had a site tag from a filesystem backup created while DMF 3.1 or later was running to a system running a pre-3.1 version of DMF is not recommended, because the attribute will appear invalid to the pre-3.1 version of DMF.

To ensure that there are no nonzero site tags, do the following:

1. While DMF is running, execute the following script to clear all site tags in managed filesystems:

   # **/usr/lib/dmf/support/dmcleartag**

   This command can take some time to run. If there are other DMF requests active for files whose site tags must be cleared, the request to clear the site tag may be queued behind the other request.

2. If the dmcleartag script completed without errors, stop DMF.

3. It is possible that a site tag was set on a file while the dmcleartag script was running, and so there may still be files with nonzero site tags. To verify that there are no nonzero site tags in the managed filesystems, run the following script:

   # **/usr/lib/dmf/support/dmanytag**

   The script will print a message to stderr if any nonzero site tags are found. If any are found, restart DMF, and repeat step 1. Otherwise, proceed to step 4.

4. Site tags may also be put on files in the DCM or disk MSP STORE_DIRECTORY. The dmcleartag script run in step 1 will clear the site tags on many of these files. However, if there are any soft-deleted files in the DCM or disk MSP STORE_DIRECTORY that have a non-zero site tag, they must be handled while the DMF daemon is not running. Run the following script to clear the tags on soft-deleted DCM MSP copies while dmfdaemon is stopped:

   # **/usr/lib/dmf/support/dmcleardcmtag**

The DMF attributes should now be in a proper state for running a previous version of DMF.

Downgrading and the Partial-State File Feature Introduced in DMF 3.2

DMF 3.2 introduced the partial-state file feature. Partial-state (PAR) files are not handled by earlier versions of DMF. If customers have installed and run DMF 3.2 or later and then wish to run an earlier version of DMF (pre-DMF 3.2), they must ensure that there are no partial-state files in the managed filesystems before installing the earlier version of DMF. Failure to do this will cause errors when running the earlier version of DMF.

Follow these steps to ensure that there are no partial-state files:

1. While DMF 3.2 is running, execute the following script to change all partial-state files in managed filesystems to be offline:

   # **/usr/lib/dmf/support/dmclearpartial**

   This command may take some time to run. If there are other DMF requests active for the partial-state files, the request to make them offline may be queued behind the other request.

2. If the `dmclearpartial` script completed without errors, stop DMF.

3. It is possible that a file was changed to partial-state while the `dmclearpartial` script was running, and so there may still be partial-state files. Verify that there are no partial-state files in the managed filesystems by running the following script:

   # **/usr/lib/dmf/support/manypartial**

   This script will print a message to `stderr` if any partial-state files are found. If any are found, restart DMF and repeat step 1. Otherwise, proceed to step 4.

4. The partial-state files should now be offline and in a proper state for running a previous version of DMF. If you are installing a version of DMF prior to DMF 3.1, you must also ensure that there are no site tags on managed files. See the instructions below.

Note: While site tags are being cleared, it is possible that files will be made partial-state. Before running a version of DMF prior to DMF 3.1, check (while DMF is stopped) both that there are no partial-state files and that there are no files with site tags.

`dmaudit`(8) Changes in DMF 3.2

The format of some of the files that `dmaudit` writes changed in DMF 3.2. The DMF 3.2 or later version of `dmaudit` is unable to read the files written by pre-DMF 3.2 versions of `dmaudit`. This means that after upgrading DMF to version 3.2 or later from a pre-DMF 3.2 version, the first time you use `dmaudit`, you must select the `snapshot` option before you can use the `inspect` option.

Logfile Changes in DMF 3.2

A change was made in DMF 3.2 to the way that the DMF daemon and the library server (LS) and MSPs refer to the daemon request number. This change should make it easier for administrators to extract all of the pertinent messages from the `SPOOL_DIR` logs for a particular request.

In previous releases of DMF, the string `Req=`*xxx* could be used to extract some log messages for daemon request number *xxx*, but there were some messages in the form `Req=`*xxx*`/`*nnn* that would not be found (such as by using the `grep`(1) command) with a pattern of `Req=`*xxx*.

A change was made to standardize all daemon and LS/MSP log messages to use the form `Req=`*xxx* for all messages. As a result, a log message formerly of the form `Req=`*xxx*`/`*nnn* would now take the form `Req=`*xxx*`,`*nnn* so as to be visible via the `grep` pattern `Req=`*xxx*. If your site uses these patterns to search DMF `SPOOL_DIR` logs, please be advised of this change and update any scripts or procedures accordingly.

Possible DMF Database Lock Manager Incompatibility On Upgrades as of DMF 3.8.3

The DMF 3.8.3 version of DMF introduced decreased DMF database lock manager delays when processes are making simultaneous lock requests. This code also introduced a backwards incompatibility between pre-3.8.3 `dmlockmgr` processes and post-3.8.3 `dmlockmgr` clients. If DMF is stopped (as recommended) via `/etc/init.d/dmf stop` immediately before installing DMF 3.8.3 or later (in a non-HA environment), there will be no incompatibility.[1]

If, however, one of the DMF administrator commands (`dmdadm`, `dmvoladm`, or `dmcatadm`) is executed after DMF has been stopped and DMF 3.8.3 or later is installed, new `dmlockmgr` clients will hang when trying to request database locks from an older version of `dmlockmgr` that was executing as the result of the administrator command.

For this reason, it is important to make sure that DMF, including the `dmlockmgr` process, is stopped via `/etc/init.d/dmf stop` immediately before installing DMF 3.8.3 or later even if the DMF daemon is not running, if you are upgrading from a pre-3.8.3 version of DMF.

1 In an HA environment, you must first remove HA control of the resource group before stopping DMF and the mounting service. See the *High Availability Guide for SGI InfiniteStorage*

Using dmmaint to Install Licenses and Configure DMF

Note: The dmmaint command is deprecated and will be removed in a future release. The preferred tool is DMF Manager; see "Configuring DMF with DMF Manager" on page 210.

On DMF servers, you can use dmmaint to install your DMF licenses and edit the DMF configuration file. The advantage to using dmmaint rather than a text editor such as vi is that you can edit the configuration file, verify your changes, and apply your changes atomically.

This appendix discusses the following:

- "Overview of dmmaint" on page 789
- "Installing the DMF License" on page 791
- "Using dmmaint to Define the Configuration File" on page 791

Overview of dmmaint

To use the dmmaint graphical user interface (GUI), ensure that your DISPLAY environment variable is defined, and then enter the following command:

```
# /usr/sbin/dmmaint &
```

Note: If DISPLAY is not defined, dmmaint reverts to line mode, which has menu selections that are equivalent to the fields and buttons on the graphic user interface. Line mode is provided for remote log in but is not recommended for general use.

The GUI displays the installed version of DMF. The **Help** menu provides access to the dmmaint and dmf.conf man pages.

The GUI buttons are as follows:

Button	Description
Configure	Lets you customize the DMF configuration file for the selected version of DMF. If this is the first time you have configured DMF, a window appears telling you that there is no configuration file. You are asked which file you would like to use as a basis for the new configuration. You may choose an existing file or one of several sample files that are preconfigured for different types of media-specific process (MSP) or the library server (LS). See "Use Sample DMF Configuration Files" on page 101. If a configuration file exists, a window appears that asks if you would like to modify the existing file or use an alternate file. If you choose an alternate file, you see the same window that you would see if this were a new configuration. After you choose a file to use as a basis, an editing session is started (in a new window) that displays a copy of that configuration file. You can make changes as desired. After exiting from the editor, you are prompted for confirmation before the original configuration file is replaced with the edited copy. For more information on configuration parameters, see Chapter 7, "DMF Configuration File" on page 271, and the `dmf.conf`(5) man page (available from the **Help** button).
Inspect	Runs the `dmcheck`(8) program to report errors. You should run this program after you have created a configuration file. If there are errors, you can click the **Configure** button, make changes, and continue to alternate between **Configure** and **Inspect** until you are satisfied that the configuration is correct.
Release Note	This button displays the DMF release note that is installed in `/usr/share/doc/packages/sgi-issp-`*ISSPversion*`/README_DMF.txt`

License Info	Displays the hostname and unique system identifier (which you need to obtain a DMF server license), the name of the license file, and a short description of the state of any DMF license within the file.
Update License	Lets you make changes to the license file. An editing session is started in a new window displaying a copy of the contents of the license file. You can add or delete licenses as desired. After you exit the editor, positive confirmation is requested before the original license file is replaced by the modified copy. For more information, see Chapter 2, "DMF Licensing" on page 67.

Installing the DMF License

To install the DMF license, do the following:

1. Select **Dependencies** to read about the hardware and software requirements that must be fulfilled before running DMF.
2. If needed, select the **Update License** button and use the mouse to copy and paste your license into the file. Close the window. Select **License Info** and examine the output to verify that the license is installed correctly.

Using `dmmaint` to Define the Configuration File

To use `dmmaint` to configure DMF, do the following:

1. Select **Configure** to edit the configuration file.
2. Click the **Inspect** button, which runs `dmcheck` to report any errors in the configuration. If there are errors, you can click the **Configure** button, make changes, and continue to alternate between **Configure** and **Inspect** until you are satisfied that the configuration is correct.

If you do not want DMF to be automatically started and stopped, see "Starting and Stopping the DMF Environment" on page 181.

Glossary

accelerated access to first byte

A partial-state file feature capability that allows you to access the beginning of an offline file before the entire file has been recalled.

active database entry

A daemon database entry whose BFID points to a valid file in the filesystem. See also *BFID* and *soft-deleted database entry*.

active metadata server

A CXFS server-capable administration node chosen from the list of potential metadata servers. There can be only one active metadata server for any given filesystem. See also *metadata*.

active parallel data-mover node

A parallel data mover node that has been enabled using `dmnode_admin`(8), has not exceeded the number of parallel data-mover node licenses on the DMF server, and is connected to the `dmnode` service on the DMF server. See also *parallel data-mover node* and *parallel data-mover node license*.

`ADMDIR_IN_ROOTFS`

The list of DMF administrative and store directories that can reside in the root (`/`) filesystem. See "`base` Object Parameters" on page 278.

`ADMIN_EMAIL`

The e-mail address to receive output from administrative tasks. See "`base` Object Parameters" on page 278.

administrative directories

See *DMF administrative directories*.

AG

See *allocation group*.

AGE_WEIGHT

A floating-point constant and floating-point multiplier to use when calculating the weight given to a file's age (for MSP/VG filesystem). See "File Weighting Parameters for a Managed Filesystem" on page 371.

AGGRESSIVE_HVFY

The parameter that specifies whether or not DMF will set the hvfy flag on volumes in the VOL database for an expanded set of error conditions. See "drivegroup Object Parameters" on page 397.

ALERT_RETENTION

Specifies the age of alert records that will be kept. See "taskgroup Object Parameters" on page 312.

ALGORITHM

The resource scheduling algorithm to be used. See "resourcescheduler Object Parameters" on page 432.

allocation group

(AG) A pool of volumes that can be transferred to a VG as needed and are returned to the pool when empty, subject to VG configuration parameters. The ALLOCATION_GROUP parameter defines a pool of volumes that have been assigned to the AG via the dmvoladm(8) command. Normally, one allocation group is configured to serve multiple VGs.

ALLOCATION_GROUP

The parameter that defines the allocation group (AG) that serves as a source of additional volumes if a VG runs out of volumes. See "volumegroup Object Parameters" on page 411.

allocationgroup

The optional configuration object that defines the VOL_MSG_TIME parameter (required only to change the default setting). See "allocationgroup Object Parameters" on page 435.

ALLOCATION_MAXIMUM

The maximum size in number of volumes to which a VG can grow by borrowing volumes from its allocation group. See "volumegroup Object Parameters" on page 411.

ALLOCATION_MINIMUM

The minimum size in number of volumes to which a VG can shrink by returning volumes to its allocation group. See "volumegroup Object Parameters" on page 411.

alternate media

The media onto which migrated data blocks are stored, usually tapes.

ARC

Scality Advanced Resilience Configuration, which uses erasure code technology to provide redundancy with reduced overhead for large objects

archive filesystem

A POSIX filesystem (such as Lustre) configured in a filesystem object in the DMF configuration file to have a MIGRATION_LEVEL of archive. This filesystem cannot be DMAPI-enabled (that is, it cannot be mounted with the dmi mount option). You can efficiently copy files between an archive filesystem to DMF secondary storage via the dmarchive(1) command.

automated space management

The combination of utilities that allows DMF to maintain a specified level of free space on a filesystem through automatic file migration.

BANDWIDTH_MULTIPLIER

(OpenVault only) A floating point number used to adjust the amount of bandwidth that the LS assumes a drive in the DG will use. See "drivegroup Object Parameters" on page 397.

base **object**

The configuration object that defines the file pathname and size parameters necessary for DMF operation. See "base Object Parameters" on page 278.

basic DMF

DMF without the Parallel Data-Mover Option.

BFID

A unique identifier, assigned to each file during the migration process, that links a migrated file to its data on alternate media.

BFID set

The collection of database entries and the file associated with a particular bit-file identifier.

BFID-set state

The sum of the states of the components that constitute a bit-file identifier set: the file state of any file and the state of any database entries (incomplete, complete, soft-deleted, or active).

bit-file ID

See *BFID*.

bit-file identifier

See *BFID*

block

Physical unit of I/O to and from media. The size of a block is determined by the type of device being written. A block is accompanied by a header identifying the chunk number, zone number, and its position within the chunk.

`BLOCK_SIZE`

The maximum block size to use when writing from the beginning of a volume. See "`drivegroup` Object Parameters" on page 397.

`BORROW_FROM_DG`

Specifies a comma-separated list of one or more OpenVault drive groups and maximum drives that are owned by another application (such as DMF) from which LTFS may borrow drives. See "`sgi-ltfs.conf` Parameters" on page 634.

BUFFERED_IO_SIZE

The size of I/O requests when reading from a filesystem using buffered I/O. See:

- "`filesystem` Object Parameters" on page 351
- "`msp` Object Parameters for a DCM MSP" on page 460

CACHE_AGE_WEIGHT

The floating-point constant and floating-point multiplier used to calculate the weight given to a file's age (for DCM MSP `STORE_DIRECTORY`). See "File Weighting Parameters for a DCM MSP `STORE_DIRECTORY`" on page 378.

CACHE_DIR

The directory in which the VG stores chunks while merging them from sparse volumes. See "`libraryserver` Object Parameters" on page 393.

CACHE_MEMBERS

The single VG or one or more MGs to be used as a fast-mount cache. See "`fastmountcache` Object Parameters" on page 390.

CACHE_SPACE

The amount of disk space (in bytes) that `dmatls` can use when merging chunks from sparse volumes. See "`libraryserver` Object Parameters" on page 393.

CACHE_SPACE_WEIGHT

The floating-point constant and floating-point multiplier to use to calculate the weight given to a file's size (for DCM MSP `STORE_DIRECTORY`). See "File Weighting Parameters for a DCM MSP `STORE_DIRECTORY`" on page 378.

candidate list

A list that contains an entry for each file in a filesystem eligible for migration, or for a file or range of files that are eligible to be made offline. This list is ordered from largest file weight (first to be migrated) to smallest. This list is generated and used internally by `dmfsmon`(8).

capability license

See *server capability license.*

capacity license

See *data capacity license.*

CAT record

An entry in the catalog (CAT) table of the LS database that tracks the location of migrated data on a volume. There is one CAT record for each migrated copy of a file. (If a migrated copy of a file is divided onto more than one physical media, there will be a CAT record for each portion.) See also *VOL record.*

CAT table

A table in the LS database that contains CAT records. See also *VOL table.*

`CHECKSUM_TYPE`

The type of checksum algorithm to use when writing new tapes. See "`volumegroup` Object Parameters" on page 411.

`CHILD_MAXIMUM`

The maximum number of child processes that the MSP is allowed to fork. See:

- "`msp` Object Parameters for a Cloud MSP" on page 449
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Parameters for a Disk MSP" on page 467
- "`msp` Object Parameters for an FTP MSP" on page 455

chunk

That portion of a file that fits on the current media volume. Most small files are written as single chunks. When a migrated file cannot fit onto a single volume, the file is split into chunks.

CIFS

Common Internet File System

client-only node

A node that is installed with the `cxfs_client.sw.base` software product; it does not run cluster administration daemons and is not capable of coordinating CXFS metadata. See also *server-capable administration node*.

`CLOUD_BACKEND`

Specifies the backend used by a given cloud instance, which determines how other cloud-specific parameters are interpreted. See "`msp` Object Parameters for a Cloud MSP" on page 449.

`CLOUD_BASE_PATH`

(S3 only) Specifies a string that is prefixed to the key (separated from the key by a slash *"/"* character) in all requests to S3. See "`msp` Object Parameters for a Cloud MSP" on page 449.

`CLOUD_BUCKET`

Specifies the name of the existing S3 bucket to use when storing files. See "`msp` Object Parameters for a Cloud MSP" on page 449.

`CLOUD_HOST`

Specifies the domain name or IP address of one or more machines that can serve as the cloud host. See "`msp` Object Parameters for a Cloud MSP" on page 449.

`CLOUD_PASSWORD`

Specifies the file that contains the cloud password information. See "`msp` Object Parameters for a Cloud MSP" on page 449.

`CLOUD_REDUNDANCY`

Specifies the level of data redundancy that is requested from the cloud service. See "`msp` Object Parameters for a Cloud MSP" on page 449.

`CLOUD_USER`

Specifies the cloud username information. See "`msp` Object Parameters for a Cloud MSP" on page 449.

COMMAND

The binary file to execute in order to initiate an MSP or LS. See:

- "`libraryserver` Object Parameters" on page 393
- "`msp` Object Parameters for a Cloud MSP" on page 449
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Parameters for a Disk MSP" on page 467
- "`msp` Object Parameters for an FTP MSP" on page 455

common arena

A shared-memory region where various DMF processes write configuration information and metrics about DMF if `EXPORT_METRICS` is enabled. Performance Co-Pilot, DMF Manager, `dmstat`, `dmtapestat`, and `dmarenadump` make use of the common arena.

complete daemon-database entry

An entry in the daemon database whose `path` field contains a key returned by its MSP or VG, indicating that the MSP or VG maintains a valid copy of the file.

compression

The mechanism by which data is reduced as it is written to secondary storage.

COMPRESSION_TYPE

Specifies the compression type level to be used with COPAN MAID when writing from the beginning of the volume. See "`drivegroup` Object Parameters" on page 397.

configuration object

A series of parameter definitions in the DMF configuration file that controls the way in which DMF operates. By changing the parameters associated with objects, you can modify the behavior of DMF.

configuration parameter

A string in the DMF configuration file that defines a part of a configuration object. By changing the values associated with these parameters, you can modify the behavior of

DMF. The parameter serves as the name of the line. Some parameters are reserved words, some are supplied by the site.

configuration stanza

A sequence of configuration parameters that define a configuration object.

COPAN MAID

Power-efficient long-term data storage based on an enterprise massive array of idle disks (MAID) platform.

COPAN VTL

Power-efficient long-term data storage based on an enterprise MAID platform using a virtual tape library (VTL).

`COPAN_VSNS`

A parameter that specifies that the fourth character of the volume serial number (VSN) indicates the RAID in the COPAN VTL or COPAN MAID that contains the volume. This specification applies for all VSNs in this library server. See "`libraryserver` Object Parameters" on page 393.

CXFS

Clustered XFS, a parallel-access shared clustered filesystem for high-performance computing environments.

daemon

A program that is run automatically by the system for a specific purpose.

daemon database

A database maintained by the DMF daemon. This database contains information such as the bit-file identifier, the MSP or VG name, and MSP or VG key for each copy of a migrated file.

DASD

See *direct-access storage device*.

data capacity license

One or more cumulative DMF licenses that permit DMF migration, corresponding to the amount of data that DMF is currently managing. See also *server capability license*.

data integrity validation

See *logical block protection*.

`DATA_LIMIT`

The maximum amount of data (in bytes) that should be selected for merging at one time. See "`taskgroup` Object Parameters" on page 312.

`DATABASE_COPIES`

One or more directories into which a copy of the DMF databases will be placed. See "`taskgroup` Object Parameters" on page 312.

data mover

A node running *data-mover processes* to migrate and recall data to secondary storage, either a *DMF server* or a *parallel data-mover node*.

data-mover processes

The individual processes that migrate data (using the *write child*) and recall data (using the *read child*).

data-pointer area

The portion of the inode that points to the file's data blocks.

`device` **object**

The configuration object that defines parameters for the DMF backup scripts' use of tape devices other than those defined by a DG. See "`device` Object Parameters" on page 348.

DCM MSP

The *disk cache manager* MSP is the disk MSP configured for *n*–tier capability by using a dedicated filesystem as a cache. DMF can manage the disk MSP's storage filesystem

and further migrate it to tape or MAID, thereby using a slower and less-expensive dedicated filesystem as a cache to improve the performance when recalling files.

DG

See *drive group*.

`DIRECT_IO_MAXIMUM_SIZE`

The maximum size of I/O requests when using `O_DIRECT` I/O to read from any managed filesystem or when migrating files down the hierarchy from the `STORE_DIRECTORY` of a DCM MSP. See "`base` Object Parameters" on page 278.

`DIRECT_IO_SIZE`

The size of I/O requests when reading from this filesystem using direct I/O. See:

- "`filesystem` Object Parameters" on page 351
- "`msp` Object Parameters for a DCM MSP" on page 460

Disaster-recovery bundle

A component produced during integrated backups that contains configuration information, the XFS inventory, a dump of the backup filesystem, and the disaster-recovery database (containing the DMF database entries that correspond to the backup filesystem)

`DISCONNECT_TIMEOUT`

Specifies the number of seconds after which the LS will consider a mover process to have exited if it cannot communicate with the process. See "`libraryserver` Object Parameters" on page 393.

disk backups

Traditional (nonintegrated) backups to disk using `xfsdump`. See also *integrated backups* and *tape backups*.

disk cache

Data on secondary storage.

disk cache manager

See *DCM MSP*.

DMAPI

Data Management Application Programming Interface.

DMF administrative and store directories

The set of directories in which DMF stores databases, log and journal files, and temporary files. The DMF configuration file specifies these directories using the following parameters:

```
HOME_DIR
JOURNAL_DIR
SPOOL_DIR
TMP_DIR
MOVE_FS
CACHE_DIR
```

`dmdaemon` **object**

The configuration object that defines parameters necessary for `dmfdaemon`(8) operation. See "`dmdaemon` Object Parameters" on page 290.

DMF daemon

The program that accepts requests to migrate data, communicates with the operating system kernel in order to maintain a file's migration state, determines the destination of migrated data, and requests the return of offline copies.

DMF direct archiving

The DMF feature that lets users manually archive files from an unmanaged POSIX filesystem directly to secondary storage via the `dmarchive`(1) command. See "Direct Archiving" on page 13.

DMF server

A node running the required DMF server software that provides DMF administration, configuration, and data mover functionality. (When using the Parallel DMF, data mover functionality is optional on the DMF server.) See *Parallel Data-Mover Option*.

DMF state

See *file state*.

DMMIGRATE_MINIMUM_AGE

The minimum file age to migrate in minutes (the `dmmigrate -m` *minutes* option). See "`taskgroup` Object Parameters" on page 312.

DMMIGRATE_MIN_OFFLINE_AGE

The value used for the `-p` option (in minutes) when `dmmigrate` is executed by the `run_dmmigrate.sh` task. This value is a file's minimum age in order for its data blocks to be released, making the file offline. See "`taskgroup` Object Parameters" on page 312.

DMMIGRATE_MIN_OFFLINE_SIZE

The value used for the `-s` option (in bytes) when `dmmigrate` is executed by the `run_dmmigrate.sh` task. This value is a file's minimum size in order for its data blocks to be released, making the file offline. See "`taskgroup` Object Parameters" on page 312.

DMMIGRATE_QUEUE_MAX

The parameter that specifies the maximum number of `dmmigrate` requests that can be outstanding in the DMF daemon queue at any given time when `dmmigrate` is executed by the `run_dmmigrate.sh` task; when this threshold is reached, the command waits before issuing more requests. See "`taskgroup` Object Parameters" on page 312.

DMMIGRATE_TRICKLE

The parameter that specifies whether or not `dmmigrate` limits the rate at which an individual `dmmigrate` command issues requests so that it will not dominate the DMF daemon (that is, the same behavior as using the `dmmigrate -t` option). See "`taskgroup` Object Parameters" on page 312.

DMMIGRATE_UNACK

The maximum number of unacknowledged requests that can be outstanding at any given time in. See "`taskgroup` Object Parameters" on page 312.

DMMIGRATE_VERBOSE

The parameter that specifies whether or not `dmmigrate` will display how many files and bytes are migrating (the `dmmigrate -v` option). See "`taskgroup` Object Parameters" on page 312.

DMMIGRATE_WAIT

The parameter that specifies whether or not `dmmigrate` will wait for all migrations to complete before exiting (the `dmmigrate -w` option). See "`taskgroup` Object Parameters" on page 312.

DR bundle

Disaster-recovery bundle

drive

A hardware device that reads and writes data to media.

drive group

(DG) One of the components of an LS. The drive group is responsible for the management of a group of interchangeable drives located in the library. These drives can be used by multiple VGs and by non-DMF processes, such as backups and interactive users. The main tasks of the DG are to monitor I/O for errors, to attempt to classify them (as volume, drive, or mounting service problems), and to take preventive action. When this document refers to *DG*, it indicates the *DMF drive group*. See also *OpenVault drive group*.

`drivegroup` **object**

The configuration object that defines a DG, one for each pool of interchangeable drives in a single library. See "`drivegroup` Object Parameters" on page 397.

DRIVE_GROUPS

One or more DGs containing drives that the LS can use for mounting and unmounting volumes. See "`libraryserver` Object Parameters" on page 393.

DRIVE_MAXIMUM

The maximum number of drives that the DG or an individual VG is allowed to attempt to use simultaneously. See:

- "drivegroup Object Parameters" on page 397
- "volumegroup Object Parameters" on page 411

DRIVE_SCHEDULER

The resource scheduler that the DG should run for the scheduling of drives. See:

- "drivegroup Object Parameters" on page 397
- "volumegroup Object Parameters" on page 411

DRIVES_TO_DOWN

An integer value that controls the number of "bad" drives the DG is allowed to try to configure down. See "drivegroup Object Parameters" on page 397.

DRIVETAB

This optional parameter provides the name of a file that is used with the tsreport --drivetab option, which causes the run_daily_drive_report and run_daily_tsreport output to contain the more readable drive name instead of the device name. See "taskgroup Object Parameters" on page 312.

DSK_BUFSIZE

The transfer size in bytes used when reading from and writing to files within the disk MSP's STORE_DIRECTORY. See:

- "msp Object Parameters for a DCM MSP" on page 460
- "msp Object Parameters for a Disk MSP" on page 467

DUALRESIDENCE_TARGET

The percentage of DCM MSP cache capacity that DMF maintains as a reserve of dual-state files whose online space can be freed if free space reaches or falls below FREE_SPACE_MINIMUM (for DCM MSP STORE_DIRECTORY). See:

- "Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376
- "msp Object Parameters for a DCM MSP" on page 460

dual-resident file

A file whose data resides online and offline in both in cache and tape/MAID (analogous to a *dual-state file*), for DMF using a DCM MSP.

dual-state file

A file whose data resides both online and offline.

DUL

See *dual-state file*

DUMP_COMPRESS

(disk and integrated backups only) The compression type and level to be used with backups. See:

- "dumpgroup Object Parameters" on page 343
- "filesystem Object Parameters" on page 351
- "taskgroup Object Parameters" on page 312

DUMP_CONCURRENCY

(disk and integrated backups only) The maximum number of filesystems that will be dumped simultaneously. See "taskgroup Object Parameters" on page 312.

DUMP_DATABASE_COPY

(third-party backups only) The path to a directory where a snapshot of the DMF databases will be placed when do_predump.sh is run. See "DUMP_DATABASE_COPY Configuration File Parameter in the taskgroup Object" on page 768.

DUMP_DESTINATION

(disk and integrated backups only) The directory in which to store backups. See "taskgroup Object Parameters" on page 312.

DUMP_DEVICE

(tape backups only) The name of the DG in the configuration file that defines how to mount the tapes that the backup tasks will use. See "taskgroup Object Parameters" on page 312.

DUMP_DR_EMAIL

(integrated backups only) Specifies one or more email addresses to which the disaster-recovery bundle is sent. See "taskgroup Object Parameters" on page 312.

DUMP_DR_EMAIL_MAX_SIZE

(integrated backups only) Specifies the maximum size of the disaster-recovery bundle that will be attached to the email sent to the addresses specified by DUMP_DR_EMAIL. See "taskgroup Object Parameters" on page 312.

DUMP_DR_MINIMAL

(integrated backups only) Specifies whether or not the backup process produces a disaster-recovery bundle of the smallest possible size. See "taskgroup Object Parameters" on page 312.

DUMP_DR_MIRRORS

(integrated backups only) Specifies one or more directories in which to place a copy of the disaster-recovery bundle. See "taskgroup Object Parameters" on page 312.

DUMP_FILE_SYSTEMS

(disk, tape, and integrated backups) One or more filesystems to back up. See:

- "dumpgroup Object Parameters" on page 343
- "taskgroup Object Parameters" on page 312

DUMP_FLUSH_DCM_FIRST

(disk, tape, and integrated backups) Specifies whether or not the dmmigrate command is run before the backups are done to ensure that all non-dual-resident files in the DCM MSP caches are migrated to tape/MAID. See:

- "msp Object Parameters for a DCM MSP" on page 460
- "taskgroup Object Parameters" on page 312

DUMP_GROUPS

(integrated backups only) Specifies a list of dumpgroup objects separated by spaces. See "taskgroup Object Parameters" on page 312.

dumpgroup **object**

(integrated backups only) The configuration object that defines dump parameters that apply to a specific subset of filesystems.

DUMP_INVENTORY_COPY

*(tape backups only)*The pathnames of one or more directories into which are copied the XFS inventory files for the backed-up filesystems. See "taskgroup Object Parameters" on page 312.

DUMP_MAX_FILESPACE

(disk, tape, and integrated backups) The maximum disk space used for files to be dumped, which may be larger or smaller than the length of the file. See:

- "dumpgroup Object Parameters" on page 343
- "filesystem Object Parameters" on page 351
- "taskgroup Object Parameters" on page 312

DUMP_MIGRATE_FIRST

(disk, tape, and integrated backups) The parameter that specifies whether or not the dmmigrate command is run before the backups are done to ensure that all

migratable files in the managed filesystems are migrated, thus reducing the amount of media space needed for the dump and making it run much faster. See:

- "`dumpgroup` Object Parameters" on page 343
- "`filesystem` Object Parameters" on page 351
- "`taskgroup` Object Parameters" on page 312

`DUMP_MIRRORS`

(disk and integrated backups only) One or more directories in which to place a copy of nonintegrated disk backups. See "`taskgroup` Object Parameters" on page 312.

`DUMP_MISC_LOCATIONS`

(integrated backups only) Specifies one or more full pathnames that must be included in the dump. See "`taskgroup` Object Parameters" on page 312.

`DUMP_POST_BACKUP_SCRIPT`

(integrated backups only) Specifies the full pathname to a user script that is run just before the disaster-recovery bundle is created (after the filesystem dumps, database snapshots, and so on are complete) . See "`taskgroup` Object Parameters" on page 312.

`DUMP_POST_DR_SCRIPT`

(integrated backups only) Specifies the full pathname to a user script that is run just after the disaster-recovery bundle is created. See "`taskgroup` Object Parameters" on page 312.

`DUMP_PRE_BACKUP_SCRIPT`

(integrated backups only) Specifies the full pathname to a user script that is run just before the backup is performed. See "`taskgroup` Object Parameters" on page 312.

`DUMP_RETENTION`

*(disk, tape, and integrated backups)*The length of time that the backups of the filesystem will be kept before the backup space is reused. See "`taskgroup` Object Parameters" on page 312.

DUMP_STREAMS

(disk and integrated backups only) The number of `xfsdump` streams (threads) to use when backing up a filesystem. See:

- "`dumpgroup` Object Parameters" on page 343
- "`filesystem` Object Parameters" on page 351
- "`taskgroup` Object Parameters" on page 312

DUMP_TAPES

(tape backups only) The path of a file that contains VSNs, one per line, for the backup tasks to use. See "`taskgroup` Object Parameters" on page 312.

DUMP_VSNS_USED

(tape backups only) A file in which the VSNs of tapes that are used are written. See "`taskgroup` Object Parameters" on page 312.

DUMP_WORKSPACE

(disk and integrated backups only) Specifies a local directory used to store temporary files created during a dump process before moving them to the DUMP_DESTINATION filesystem. See "`taskgroup` Object Parameters" on page 312.

DUMP_XFSDUMP_PARAMS

(disk, tape, and integrated backups) Passes parameters to the `xfsdump` program.

- "`dumpgroup` Object Parameters" on page 343
- "`filesystem` Object Parameters" on page 351
- "`taskgroup` Object Parameters" on page 312

EOT

End-of-volume marker (historically known as *EOT* for *end-of-tape*)

EXPORT_METRICS

Enables DMF's use of the common arena for use by Performance Co-Pilot (PCP), dmstat(8), dmarenadump(8), and other commands. See "base Object Parameters" on page 278.

EXPORT_QUEUE

Instructs the daemon to export details of its internal request queue to *SPOOL_DIR*/daemon_exports every two minutes, for use by dmstat(8) and other utilities. See "dmdaemon Object Parameters" on page 290.

FADV_SIZE_MAID

Specifies when to call posix_fadvise() with advice POSIX_FADV_DONTNEED for COPAN MAID volumes. See "drivegroup Object Parameters" on page 397.

FADV_SIZE_MSP

Specifies the size of files in the MSP's STORE_DIRECTORY for which posix_fadvise() will be called with advice POSIX_FADV_DONTNEED. See:

- "msp Object Parameters for a DCM MSP" on page 460
- "msp Object Parameters for a Disk MSP" on page 467

fast-mount cache

A migration target with fast mount/position characteristics (such as COPAN MAID) that is used in conjunction with another target (such as physical tape). In a fast-mount cache configuration, DMF simultaneously migrates data to a copy on the cache target and to secondary–storage copies on the other targets. This configuration provides similar functionality to a DCM MSP but does not migrate data from the cache to tier-3, so volumes on the cache can be freed immediately when the fullness threshold is reached and volume merging is avoided. A taskgroup object is required to empty full volumes in the cache after a specified threshold is reached.

FC

Fibre Channel.

fhandle

See *file handle*.

file

An inode and its associated data blocks; an empty file has an inode but no data blocks.

file handle

The DMAPI identification for a file. You can use the `dmscanfs`(8), `dmattr`(1), and `dmfind`(1) commands to find file handles.

file state

The migration state of a file as indicated by the `dmattr`(1) command. A file can be regular (not migrated), migrating, dual-state, offline, partial-state, unmigrating, never-migrated, or have an invalid DMF state.

file tag

A site-assigned 32-bit integer associated with a specific file, allowing the file to be identified and acted upon.

`FILE_RETENTION_DAYS`

The age (in days) of a file that must not be deleted from the fast-mount cache. See "`taskgroup` Object Parameters" on page 312.

`filesystem` **object**

The configuration object that defines parameters necessary for migrating files in that filesystem. See "`filesystem` Object Parameters" on page 351.

`FMC_MOVEFS`

The specific scratch `MOVE_FS` filesystem to be used when moving files to be retained in the fast-mount cache. See "`taskgroup` Object Parameters" on page 312.

`FMC_NAME`

The name of a `fastmountcache` object. See "`taskgroup` Object Parameters" on page 312.

FORWARD_RECALLS

The parameter that specifies whether or not a recall should be directed to another VG or MSP if the volume required for the recall is busy because it is being written to. See "volumegroup Object Parameters" on page 411.

FREE_DUALRESIDENT_FIRST

Specifies whether dmdskfree will first free dual-resident files before freeing files it must migrate (for DCM MSP STORE_DIRECTORY). See "Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376.

FREE_DUALSTATE_FIRST

Specifies whether or not dmfsfree will first free dual-state and partial-state files before freeing files it must migrate (for MSP/VG filesystem). See "Automated Space-Management Parameters for a Managed Filesystem" on page 369.

FREE_SPACE_DECREMENT

The percentage of filesystem space by which dmfsmon or dmdskmsp will decrement FREE_SPACE_MINIMUM (if it cannot find enough files to migrate) so that the value is reached. The decrement is applied until a value is found that dmfsmon can achieve. See:

- "Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376
- "Automated Space-Management Parameters for a Managed Filesystem" on page 369

FREE_SPACE_MINIMUM

The minimum integer percentage of the total filesystem space that dmfsmon tries to maintain as free. See:

- "Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376
- "Automated Space-Management Parameters for a Managed Filesystem" on page 369

FREE_SPACE_TARGET

The integer percentage of total filesystem space that dmfsfree or dmdskfree tries to maintain as free if free space reaches or falls below the FREE_SPACE_MINIMUM threshold. See:

- "Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376
- "Automated Space-Management Parameters for a Managed Filesystem" on page 369

FREE_VOLUME_MINIMUM

The minimum percentage of free volumes in the COPAN MAID or COPAN VTL fast-mount cache that will cause run_fmc_free.sh to begin freeing full volumes in order to meet the percentage set for FREE_VOLUME_TARGET. See "taskgroup Object Parameters" on page 312.

FREE_VOLUME_TARGET

The percentage of volumes in the COPAN MAID or COPAN VTL fast-mount cache that run_fmc_free.sh tries to free when the FREE_VOLUME_MINIMUM threshold is reached. See "taskgroup Object Parameters" on page 312.

freed file

A file that has been migrated and whose data blocks have been released.

FTP

File Transfer Protocol

FTP_ACCOUNT

The account ID to use when migrating files to the remote system. See "msp Object Parameters for an FTP MSP" on page 455.

FTP_COMMAND

Additional commands to send to the remote system. See "msp Object Parameters for an FTP MSP" on page 455.

FTP_DIRECTORY

The directory to use on the remote system. See "msp Object Parameters for an FTP MSP" on page 455.

FTP_HOST

The Internet hostname of the remote machine on which files are to be stored. See "msp Object Parameters for an FTP MSP" on page 455.

FTP MSP

The daemon-like media-specific process (MSP) that copies data blocks onto alternate media and assigns keys to identify the location of the migrated data using the File Transfer Protocol (FTP) to transfer to and from disks of another system on the network.

FTP_PASSWORD

The file containing the password to use when migrating files to the remote system. This file must be owned by root and be only accessible by root. See "msp Object Parameters for an FTP MSP" on page 455.

FTP_PORT

The port number of the FTP server on the remote system. See "msp Object Parameters for an FTP MSP" on page 455.

FTP_USER

The user name to use when migrating files to the remote system. See "msp Object Parameters for an FTP MSP" on page 455.

FULL_ZONE

The parameter that limits when requests are flushed to a mounted volume that is not writing to a zone. See "volumegroup Object Parameters" on page 411.

FULL_THRESHOLD_BYTES

The parameter that determines whether or not the disk MSP will tell the DMF daemon when it is full. See "msp Object Parameters for a Disk MSP" on page 467.

GET_WAIT_TIME

The parameter that limits the amount of time (in seconds) that DMF will continue writing to a volume after receiving a recall request for that volume. See "`volumegroup` Object Parameters" on page 411.

GROUP_MEMBERS

The list of VGs and/or MSPs that will be a member of the migrate group. See "`migrategroup` Object Parameters" on page 427.

GUARANTEED_DELETES

The number of child processes that are guaranteed to be available for processing delete requests. See:

- "`msp` Object Parameters for a Cloud MSP" on page 449
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Parameters for a Disk MSP" on page 467
- "`msp` Object Parameters for an FTP MSP" on page 455

GUARANTEED_GETS

The number of child processes that are guaranteed to be available for processing `dmget`(1) requests. See:

- "`msp` Object Parameters for a Cloud MSP" on page 449
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Parameters for a Disk MSP" on page 467
- "`msp` Object Parameters for an FTP MSP" on page 455

HA

High availability

HA resource

A service, associated with an IP address, that is managed by SUSE Linux Enterprise High Availability Extension (HA). Also see *resource* for DMF Manager.

HAE

SUSE Linux Enterprise High Availability Extension.

hard-delete

The action of removing a soft-deleted database entry. See also *soft-deleted database entry*.

HBA_BANDWIDTH

(OpenVault only) The I/O bandwidth capacity of an HBA port that is connected to tape drives on a node. See:

- "base Object Parameters" on page 278
- "node Object Parameters" on page 294

HFREE_TIME

The minimum number of seconds that a tape no longer containing valid data must remain unused before the VG overwrites it. See "volumegroup Object Parameters" on page 411.

HOME_DIR

The base pathname for directories in which DMF databases and related files reside. See "base Object Parameters" on page 278.

HTML_REFRESH

The refresh rate (in seconds) of the generated HTML pages. See "resourcewatcher Object Parameters" on page 434.

HTTPS

Hypertext Transfer Protocol Secure

IMMUTABLE_BFIDS

Specifies whether or not DMF should restrict operations that change BFIDs of the migrated files in this filesystem. See "filesystem Object Parameters" on page 351.

IMPORT_DELETE

A parameter that specifies whether the MSP should honor hard-delete requests from the DMF daemon. See:

- "msp Object Parameters for a Disk MSP" on page 467
- "msp Object Parameters for an FTP MSP" on page 455

IMPORT_ONLY

A parameter that specifies whether the VG/MSP is used only for recalling files. See:

- "msp Object Parameters for a Disk MSP" on page 467
- "msp Object Parameters for an FTP MSP" on page 455
- "volumegroup Object Parameters" on page 411

incomplete daemon-database entry

An entry in the daemon database for an MSP or VG that has not finished copying the data, and therefore has not yet returned a key. The path field in the database entry is NULL.

incompletely migrated file

A file that has begun the migration process, but for which one or more copies on alternate media have not yet been made.

inode

The portion of a file that contains the bit-file identifier, the state field, and the data pointers.

integrated backups

(integrated backups) The backup method using xfsdump that copies backup data to a filesystem that is managed by DMF. See also *disk backups*, *tape backups*, *third-party backups*.

integrated data-mover functionality

The ability of the DMF server to move data. See also *parallel data-mover node*.

INTERFACE

The IP address or associated name of this node to be used for communication between DMF components. See "node Object Parameters" on page 294.

IP

Internet Protocol

JOURNAL_DIR

The base pathname for directories in which the daemon database and LS journal files will be written. See "base Object Parameters" on page 278.

JOURNAL_RETENTION

The length of time to keep journals. See "taskgroup Object Parameters" on page 312.

JOURNAL_SIZE

The maximum size (in bytes) of the database journal file before DMF closes it and starts a new file. See "base Object Parameters" on page 278.

KILL_WC_IF_DEMAND

The parameter that specifies whether or not an idle write child will be terminated immediately if there is demand for the drive.

LABEL_TYPE

The label type used when writing volumes from the beginning. See "drivegroup Object Parameters" on page 397.

LCP

Library control program

library server

(LS) A daemon-like process by which data blocks are copied onto secondary storage and that maintains the location of the migrated data. Each LS has an associated LS database with catalog (CAT) and volume (VOL) records. An LS can be configured to contain one or more DGs.

`libraryserver` **object**

The configuration object that defines parameters relating to a tape library for an LS. See "`libraryserver` Object Parameters" on page 393.

`LICENSE_FILE`

The full pathname of the file containing the license used by DMF. See "`base` Object Parameters" on page 278.

`LOG_RETENTION`

Specifies the age of files that will be kept when the `run_remove_logs.sh` task is run. See "`taskgroup` Object Parameters" on page 312.

logical block protection

A mechanism provided by tape drive manufactures to provide a checksum for data validation at the end of each tape block (also known as *data integrity validation*).

`LOGICAL_BLOCK_PROTECTION`

Specifies whether *logical block protection* should be turned on when reading and writing tapes. See "`volumegroup` Object Parameters" on page 411.

LS

See *library server*.

LS database

The database containing catalog (CAT) and volume (VOL) records associated with a library server (LS). See also *CAT record* and *VOL record*.

`LS_NAMES`

The library servers used by the DMF daemon. See "`dmdaemon` Object Parameters" on page 290.

`LTFS_CART_GROUP`

Specifies the single OpenVault cartridge group to be reserved for use by LTFS. The default is `ltfs_carts`. See "`sgi-ltfs.conf` Parameters" on page 634.

LTFS_DRIVE_GROUP

Specifies the single OpenVault drive group to be reserved for use by LTFS. See "sgi-ltfs.conf Parameters" on page 634.

LTFS_MAX_DRIVES

Specifies the maximum number of drives that can be in the drive group specified by LTFS_DRIVE_GROUP at any given time. See "sgi-ltfs.conf Parameters" on page 634.

LTFS_MOUNT_ROOT

Specifies the path that will be added to the tape name to form the mount point of the LTFS filesystem. See "sgi-ltfs.conf Parameters" on page 634.

LTFS_OV_KEYFILE

Specifies the path to the keyfile used by LTFS when connecting to the OpenVault server. See "sgi-ltfs.conf Parameters" on page 634.

MAID

Massive array of idle disks.

managed filesystem

A DMAPI-mounted XFS or CXFS filesystem, configured in a filesystem object in the DMF configuration file, on which DMF can migrate or recall files. (When using Parallel DMF, it must be CXFS.) See *Parallel Data-Mover Option*.

MAX_ALERTDB_SIZE

Specifies the maximum size of the alerts database. See "taskgroup Object Parameters" on page 312.

MAX_CACHE_FILE

The largest chunk (in bytes) that will be merged using the merge disk cache. See "libraryserver Object Parameters" on page 393.

MAX_CHUNK_SIZE

The size (in bytes) of the chunk into which the VG should break up large files as it writes data to secondary storage. See "volumegroup Object Parameters" on page 411.

MAX_IDLE_PUT_CHILDREN

The maximum number of idle write child (dmatwc) processes that will be allowed simultaneously for a VG. See "volumegroup Object Parameters" on page 411.

MAX_DRIVES_PER_NODE

(This parameter has been deprecated and will be ignored.)

MAX_MANAGED_REGIONS

The maximum number of managed regions that DMF will assign to a file on a per-filesystem basis. You can set MAX_MANAGED_REGIONS to any number that is less than the actual number of regions that will fit in a filesystem attribute. See "filesystem Object Parameters" on page 351.

MAX_MS_RESTARTS

The maximum number of times DMF can attempt to restart the mounting service (TMF or OpenVault) without requiring administrator intervention. See "drivegroup Object Parameters" on page 397.

MAX_PERFDB_SIZE

Specifies the maximum size of the performance records database. See "taskgroup Object Parameters" on page 312.

MAX_PUT_CHILDREN

The maximum number of write child (dmatwc) processes that will be scheduled simultaneously for the DG or an individual VG. See:

- "drivegroup Object Parameters" on page 397
- "volumegroup Object Parameters" on page 411

media-specific process

(MSP) The daemon-like process by which data blocks are copied onto alternate media and that assigns keys to identify the location of the migrated data.

MERGE_CUTOFF

A limit at which the VG will stop scheduling tapes for merging. See "volumegroup Object Parameters" on page 411.

MERGE_INTERFACE

The IP address or associated name on this node to be used when merging sparse volumes via sockets. See "node Object Parameters" on page 294.

MERGE_THRESHOLD

The integer percentage of active data on a volume less than which DMF will consider a volume to be sparse and allow merging. See "volumegroup Object Parameters" on page 411.

merging

See *volume merging*.

MESSAGE_LEVEL

The highest message level that will be written to a log file (the higher the number, the more messages written). See:

- "dmdaemon Object Parameters" on page 290
- "filesystem Object Parameters" on page 351
- "libraryserver Object Parameters" on page 393
- "msp Object Parameters for a Cloud MSP" on page 449
- "msp Object Parameters for a DCM MSP" on page 460
- "msp Object Parameters for a Disk MSP" on page 467
- "msp Object Parameters for an FTP MSP" on page 455
- "services Object Parameters" on page 300
- Chapter 10, "Message Log Files" on page 507

metadata

Information that describes a file, such as the file's name, size, location, and permissions.

metadata server

The CXFS server-capable administration node that coordinates updating of metadata on behalf of all nodes in a cluster. There can be multiple potential metadata servers, but only one is chosen to be the active metadata server for any one filesystem.

METRICS_RETENTION

Specifies the retention time for the DMF cumulative (totals and averages) metrics. See "base Object Parameters" on page 278.

migrated file

A file that has one or more complete offline copies and no pending or incomplete offline copies.

migrate group

A logical collection of VGs or MSPs that you combine into a set in order to have a single destination for a migrate request. See `migrategroup`.

`migrategroup`

The configuration object that combines a set of VGs and MSPs so that they can be used as a single destination for a migrate request. See "`migrategroup` Object Parameters" on page 427.

migrating file

A file that has a bit-file identifier but whose offline copies are in progress.

`MIGRATION_LEVEL`

The highest level of migration service allowed. See:

- "`dmdaemon` Object Parameters" on page 290
- "`filesystem` Object Parameters" on page 351
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Parameters for a Disk MSP" on page 467

`MIGRATION_TARGET`

The integer percentage of total filesystem space that `dmfsmon` tries to maintain as a reserve of space that is free or occupied by dual-state files (whose online space can be freed quickly) if free space reaches or falls below `FREE_SPACE_MINIMUM`. See "Automated Space-Management Parameters for a Managed Filesystem" on page 369.

`MIN_ARCHIVE_SIZE`

Determines whether direct or buffered I/O is used when reading from this filesystem. See "`filesystem` Object Parameters" on page 351.

MIN_DIRECT_SIZE

Determines whether direct or buffered I/O is used when reading from this filesystem. See:

- "filesystem Object Parameters" on page 351
- "msp Object Parameters for a DCM MSP" on page 460
- open(2) man page for a description of direct I/O

MIN_VOLUMES

The minimum number of unused volumes that can exist in the LS database for this VG without operator notification. See "volumegroup Object Parameters" on page 411.

monitored filesystem

A filesystem configured in the DMF configuration file such that DMF will monitor is fullness and run user-specified tasks, but for which it will not migrate or recall data.

MOUNT_BLOCKED_TIMEOUT

The maximum number of minutes to wait for a volume to be mounted when another process is using the drive. See "drivegroup Object Parameters" on page 397.

MOUNT_SERVICE

The mounting service. See:

- "device Object Parameters" on page 348
- "drivegroup Object Parameters" on page 397

MOUNT_SERVICE_GROUP

The name by which the object's devices are known to the mounting service. See:

- "device Object Parameters" on page 348
- "drivegroup Object Parameters" on page 397

MOUNT_TIMEOUT

The maximum number of minutes to wait for a volume to be mounted. See "drivegroup Object Parameters" on page 397.

MOVE_FS

One or more scratch filesystems used by dmmove(8) to move files between MSPs or VGs. See "dmdaemon Object Parameters" on page 290.

MSG_DELAY

The number of seconds that all drives in the DG can be down before DMF sends an e-mail message to the administrator and logs an error message. See "drivegroup Object Parameters" on page 397

MSP

The media-specific process (MSP), a daemon-like process by which data blocks are copied onto alternate media and that assigns keys to identify the location of the migrated data. See also: *cloud MSP*, *disk MSP*, *DCM MSP*, *FTP MSP*, and *non-cache MSP*.

MSP database entry

The daemon database entry for a file that contains the path or key that is used to inform a particular media-specific process (MSP) where to locate the copy of the file's data.

MSP_NAMES

Names the media-specific processes (MSPs) used by the DMF daemon. See "dmdaemon Object Parameters" on page 290.

msp **object**

The configuration object that defines parameters necessary for the operation of a media-specific process. There is one msp object for each MSP. See:

- "msp Object Parameters for a DCM MSP" on page 460
- "msp Object Parameters for a Disk MSP" on page 467
- "fastmountcache Object Parameters" on page 390

MULTIPLIER

The amount of data to be sent to a group member relative to the other members listed in GROUP_MEMBERS when using the ROUND_ROBIN_BY_BYTES or

ROUND_ROBIN_BY_FILES for ROTATION_STRATEGY. See "migrategroup Object Parameters" on page 427.

MULTITAPE_NODES

(Parallel DMF and OpenVault only). The parameter that restricts the recall of a file that involves multiple tapes to one of the specified mover nodes. See "drivegroup Object Parameters" on page 397.

MVS_UNIT

The storage device type on an MVS system. See "msp Object Parameters for an FTP MSP" on page 455.

NAME_FORMAT

The substitution strings that form a template for filenames used in secondary storage. See:

- "msp Object Parameters for a Cloud MSP" on page 449
- "msp Object Parameters for a DCM MSP" on page 460
- "msp Object Parameters for a Disk MSP" on page 467
- "msp Object Parameters for an FTP MSP" on page 455

near-line storage

Storage in which tapes are mounted by robot.

NFS

Network File System

NODE_ANNOUNCE_RATE

The rate in seconds at which a node will contact the dmnode_service on the DMF server to announce its presence. See "services Object Parameters" on page 300.

NODE_BANDWIDTH

(OpenVault only) The I/O bandwidth capacity of the node. See:

- "base Object Parameters" on page 278
- "node Object Parameters" on page 294

NODE_TIMEOUT

The number of seconds after which the data mover functionality on the DMF server or on a parallel data-mover node will be considered inactive if it has not contacted the dmnode_service on the DMF server. See "services Object Parameters" on page 300.

non-cache MSP

Cloud MSP, disk MSP, or FTP MSP.

nonintegrated backups

Traditional backup methods using disk or tape that were in place prior to the introduction of the integrated-backup method. See also *integrated backups*

nonmigrated file

A file that does not have a bit-file identifier or any offline copies. See *regular file*.

offline file

A file whose inode contains a bit-file identifier but whose disk blocks have been removed. The file's data exists elsewhere in copies on alternate media.

offline pointer

In MSP and VG processing, a character string that the MSP or VG returns to the daemon to indicate how a file is to be retrieved.

OpenVault

A storage library management facility that improves how applications can manage, store, and retrieve removable media.

OpenVault drive group

A group of interchangeable devices. See also DMF *drive group*.

orphan chunk

An unused area in an LS catalog (CAT) database entry resulting from the removal of migrated files.

orphan database entry

An unused daemon database entry resulting from the removal of a migrated file during a period in which the DMF daemon is not running.

`OV_ACCESS_MODES`

(OpenVault only) The OpenVault access mode. See:

- "`device` Object Parameters" on page 348
- "`drivegroup` Object Parameters" on page 397

`OV_INTERCHANGE_MODES`

(OpenVault only) A list of interchange mode names that control how data is written to secondary storage. See:

- "`device` Object Parameters" on page 348
- "`drivegroup` Object Parameters" on page 397

`OV_KEY_FILE`

(OpenVault only) The file containing the OpenVault keys used by DMF. See "`base` Object Parameters" on page 278.

`OV_SERVER`

(OpenVault only) Specifies the name associated with the IP address on which the OpenVault server is listening. See:

- "`base` Object Parameters" on page 278
- "`sgi-ltfs.conf` Parameters" on page 634

oversubscribe

A ratio of offline space to the total amount of space for a given DMF filesystem (including space that is free, space that is occupied by regular files, space that is occupied by files that are migrated, including dual-state files.

parallel data-mover node

A node, installed with DMF data mover software and underlying CXFS client-only software, that provides dedicated data mover functionality in addition to the DMF server, increasing data throughput and enhancing resiliency.

parallel data-mover node license

A DMF license installed on the DMF server that permits one parallel data-mover node to be active when using Parallel DMF. There can be multiple licenses installed, one for each parallel data-mover node that is active at any one time. See also *parallel data-mover node* and *Parallel Data-Mover Option*.

Parallel Data-Mover Option

Optional software and licenses available for purchase that allow you to run parallel data-mover nodes in order to increase data throughput and enhance resiliency.

Parallel DMF

DMF using the Parallel Data-Mover Option. See *Parallel Data-Mover Option*.

parameter

See *configuration parameter*.

partial-state file

A file that has more than one region. DMF allows a file to include up to four distinct file regions. See also *region*.

partial-state file online retention

A partial-state file feature capability that allows you to keep a specific region of a file online while freeing the rest of it (for example, if you wanted to keep just the beginning of a file online). See also *partial-state file*.

partial-state file recall

A partial-state file feature capability that allows you to recall a specific region of a file without recalling the entire file. For more information, see the `dmput`(1) and `dmget`(1) man pages. See also *partial-state file*.

`PARTIAL_STATE_FILES`

Enables or disables the DMF daemon's ability to produce partial-state files. See "`dmdaemon` Object Parameters" on page 290.

`PENALTY`

A parameter used to reduce the priority of requests from a VG that is not the next one preferred by the round-robin algorithm. See "`resourcescheduler` Object Parameters" on page 432.

`PERF_RETENTION`

The length of time to keep performance records. See "`taskgroup` Object Parameters" on page 312.

`PERFTRACE_METRICS`

Enables or disables collection of performance tracking information from DMF. See "`base` Object Parameters" on page 278.

persistent drive reservation

The locking of a given drive to a given path until the lock is dropped, which helps protect against data loss due to rewind or write requests from other paths.

`POLICIES`

The names of the configuration objects defining policies for this filesystem. See:

- "`filesystem` Object Parameters" on page 351
- "`msp` Object Parameters for a DCM MSP" on page 460

policy

Rules that tell DMF how to determine MSP or VG selection, automated space-management policies, and/or file weight calculations.

`policy` **object**

The configuration object that specifies parameters to determine MSP or VG selection, automated space management policies, and/or file weight calculations in automated space management. See "`policy` Object" on page 364.

`POSITIONING`

How the volume should be positioned. See "`drivegroup` Object Parameters" on page 397.

`POSITION_RETRY`

The level of retry in the event of a failure during zone positioning. See "`drivegroup` Object Parameters" on page 397.

`POSIX_FADVISE_SIZE`

Specifies the number of bytes after which DMF will call `posix_fadvise()` with advice `POSIX_FADV_DONTNEED` when recalling files. See "`filesystem` Object Parameters" on page 351.

`PRIORITY_PERIOD`

Specifies the number of minutes after which a migrating file gets special treatment. See "`msp` Object Parameters for a DCM MSP" on page 460.

primary filesystem

See *managed filesystem*.

`PUT_IDLE_DELAY`

The number of seconds that an idle `dmatwc` (write child) process will be allowed to stay alive. See "`volumegroup` Object Parameters" on page 411.

`PUTS_TIME`

The minimum number of seconds a VG waits after it has requested a drive for a write child before it tells a lower priority child to go away. See "`volumegroup` Object Parameters" on page 411.

RAID

Redundant array of independent disks.

raw time

The time in seconds since January 1, 1970.

read child

A data-mover process that recalls data from tape.

READ_ERR_MAXIMUM

The maximum number of I/O errors that will be tolerated when recalling a file. See "drivegroup Object Parameters" on page 397.

READ_ERR_MINIMUM

The minimum number of I/O errors that will be tolerated when recalling a file. See "drivegroup Object Parameters" on page 397.

READ_ERR_TIMEOUT

The elapsed number of seconds since the first I/O error was seen. See "drivegroup Object Parameters" on page 397.

READ_IDLE_DELAY

The number of seconds an idle read child (dmatrc) can wait before being told to exit. See:

- "drivegroup Object Parameters" on page 397
- "volumegroup Object Parameters" on page 411

READ_TIME

The interval, in seconds, after which the VG will evaluate whether a read child should be asked to go away (even if it is in the middle of recalling a file) so that a higher priority child can be started. See "volumegroup Object Parameters" on page 411.

recall a file

To request that a migrated file's data be moved back (unmigrated) onto the filesystem disk, either by explicitly entering the `dmget`(1) command or by executing another command that will open the file, such as the `vi`(1) command.

`RECALL_NOTIFICATION_RATE`

The approximate rate, in seconds, at which regions of a file being recalled are put online. This allows for access to part of a file before the entire file is recalled. See "`dmdaemon` Object Parameters" on page 290.

`REDIRECT_TIME_DG_DOWN`

Specifies the number of seconds that DMF will wait before redirecting recalls; after all drives have been down for *REDIRECT_TIME_DG_DOWN* seconds, DMF will start forwarding every recall to the VG that holds the secondary copy. See "`drivegroup` Object Parameters" on page 397.

region

A contiguous range of bytes that have the same residency state. The range state can be migrating (`MIG`), dual-state (`DUL`), offline (`OFL`), or unmigrating (`UNM`).

regular file

A file with no bit-file identifier and no offline copies.

`REINSTATE_DRIVE_DELAY`

The number of minutes after which a drive that was configured down by the DG will be automatically reinstated and made available for use again. See "`drivegroup` Object Parameters" on page 397.

`REINSTATE_VOLUME_DELAY`

The number of minutes after which a volume that had its `HLOCK` flag set by DMF will be automatically reinstated and made available for use again. See "`drivegroup` Object Parameters" on page 397.

RELEASETHREADS

The number of threads in the DMF daemon that will be created to handle requests to release filesystem blocks when files are made offline (OFL). See "filesystem Object Parameters" on page 351.

REMALERT_PARAMS

Parameters to be executed by run_remove_alerts.sh. See "taskgroup Object Parameters" on page 312.

REMPERF_PARAMS

Parameters to be executed by run_remove_perf.sh. See "taskgroup Object Parameters" on page 312.

RESERVED_VOLUMES

Defines the number of volumes that the VG will reserve for volume merging or that will trigger selection of another volume within a migrate group. See "volumegroup Object Parameters" on page 411.

resource

A *resource* is a filesystem or hardware component used by DMF. Also see *HA resource*

resource group

A service, associated with an IP address, that is managed by SUSE Linux Enterprise High Availability Extension.

resourcescheduler **object**

The configuration object that defines parameters relating to scheduling of devices in a DG when requests from VGs exceed the number of devices available. See "resourcewatcher Object Parameters" on page 434.

resourcewatcher **object**

The configuration object that defines parameters relating to the production of files informing the administrator about the status of the LS and its components. See "resourcewatcher Object Parameters" on page 434.

REST

Representational state transfer

REWIND_DELAY

The number of seconds that an idle read child (dmatrc) can wait before rewinding. See:

- "drivegroup Object Parameters" on page 397
- "volumegroup Object Parameters" on page 411

ROTATION_STRATEGY

The method in which migration requests will rotate through the VGs and MSPs that are members of this group. See "migrategroup Object Parameters" on page 427.

ROUND_ROBIN_BY_BYTES

The ROTATION_STRATEGY value specifying that a certain number of bytes (defined by MULTIPLIER) are sent to each VG/MSP member specified in GROUP_MEMBERS. See "migrategroup Object Parameters" on page 427.

ROUND_ROBIN_BY_FILES

The ROTATION_STRATEGY value specifying that a certain number of files (defined by MULTIPLIER) are sent to each VG/MSP member specified in GROUP_MEMBERS. See "migrategroup Object Parameters" on page 427.

RRR

S3 Reduced Redundancy Storage

RUN_TASK

A DMF maintenance command to be executed. See:

- "Automated Maintenance Tasks" on page 175
- "`drivegroup` Object Parameters" on page 397
- "`libraryserver` Object Parameters" on page 393
- "`taskgroup` Object Parameters" on page 312
- "`volumegroup` Object Parameters" on page 411

SCAN_FILESYSTEMS

The parameter that specifies for the `run_filesystem_scan.sh` script the filesystems that `dmscanfs` will scan. See "`taskgroup` Object Parameters" on page 312.

SCAN_FOR_DMSTAT

The parameter that specifies for the `run_filesystem_scan.sh` script whether additional output files (`bfid2path` and/or `fhandle2bfid+path`) are created, also depending upon the setting for SCAN_PARAMS. See "`taskgroup` Object Parameters" on page 312.

SCAN_OUTPUT

The parameter that specifies for the `run_filesystem_scan.sh` script the name of the file into which `dmscanfs` will place output. See "`taskgroup` Object Parameters" on page 312.

SCAN_PARALLEL

The parameter that specifies for the `run_filesystem_scan.sh` script whether `dmscanfs` will scan filesystems in parallel. See "`taskgroup` Object Parameters" on page 312.

SCAN_PARAMS

The parameter that specifies additional `dmscanfs` parameters for the `run_filesystem_scan.sh` task. See "`taskgroup` Object Parameters" on page 312.

secondary storage

The offline media onto which file data is migrated. See also *managed filesystem*.

SELECT_LOWER_VG

Defines which VGs should maintain secondary-storage copies of files in the cache, and under what conditions that would define dual-residence. (It is not used for defining which VG to use for recalls; for that, see the definitions of the LS_NAMES, MSP_NAMES, DRIVE_GROUPS, and VOLUME_GROUPS parameters.) See "VG Selection Parameters for a DCM MSP STORE_DIRECTORY" on page 380.

SELECT_MSP

The media-specific processes (MSPs) to use for migrating a file. See "MSP/VG Selection Parameters for a Managed Filesystem" on page 374.

SELECT_VG

The volume groups (VGs) to use for migrating a file. See "MSP/VG Selection Parameters for a Managed Filesystem" on page 374.

SEQUENTIAL

The ROTATION_STRATEGY value specifying that each COPAN shelf will be filled before advancing to the next shelf.

server capability license

The DMF license that permits DMF migrations to exceed 1 TB when installed in conjunction with one or more DMF data capacity licenses. See also *data capacity licenses*.

SERVER_NAME

Hostname of the machine on which the DMF server is running (used for HA configurations or configurations using Parallel DMF). See "base Object Parameters" on page 278.

SERVICES

The name of the services object used to configure DMF services on a node when using Parallel DMF. See "node Object Parameters" on page 294.

SERVICES_PORT

The port number on which DMF starts a locator service, which DMF uses to locate other DMF services. See "services Object Parameters" on page 300.

site-defined policy

A site-specific library of C++ functions that DMF will consult when making decisions about its operation.

SITE_SCRIPT

The site-specific script to execute when dmfsfree, dmdskfree, or dmfsmon is run. See:

- "Automated Space-Management Parameters for a DCM MSP STORE_DIRECTORY" on page 376
- "Automated Space-Management Parameters for a Managed Filesystem" on page 369
- "msp Object Parameters for a DCM MSP" on page 460

SMB

Server Message Block

snapshot

The information about all bit-file identifier sets that is collected and analyzed by dmaudit(8). The snapshot analysis is available from the report function.

soft-delete

The action of adding a time stamp to the delete field of a daemon database entry. See also *active database entry* and *soft-deleted database entry*.

soft-deleted database entry

A daemon database entry whose BFID points to a file that is no longer present in the managed filesystem (because it has been modified or removed) but might still reside on backup media. See also *active database entry*, *BFID*, and *soft-delete*.

SPACE_WEIGHT

The floating-point constant and floating-point multiplier to use to calculate the weight given to a file's size (for MSP/VG managed filesystem). See "File Weighting Parameters for a Managed Filesystem" on page 371.

sparse volume

A volume containing only a small amount of active information.

special file

A device file in UNIX or Linux. (DMF never migrates special files.)

SPOOL_DIR

The base pathname for directories in which DMF log files are kept. See "base Object Parameters" on page 278.

SSL

Secure Sockets Layer

standby metadata server node

A CXFS server-capable administration node that is configured as a potential metadata server for a given filesystem, but does not currently run any applications that will use that filesystem.

state field

The field in the inode that shows the current migration state of a file.

STORE_DIRECTORY

The directory used to hold files for a DCM or disk MSP. See:

- "msp Object Parameters for a DCM MSP" on page 460
- "msp Object Parameters for a Disk MSP" on page 467

tape backups

Traditional (nonintegrated) backups to tape using `xfsdump`. See also *disk backups* and *integrated backups*.

tape block

See *block*.

tape drive

See *drive*.

tape chunk

See *chunk*.

tape merging

See *volume merging*.

task

A process initiated by the DMF event mechanism. Configuration tasks that allow certain recurring administrative duties to be automated are defined with configuration parameters.

`taskgroup`

A configuration object that defines parameters necessary for the automatic completion of specific maintenance tasks. See "`dmdaemon` Object Parameters" on page 290.

TASK_GROUPS

The objects containing tasks that the daemon or LS should run. See:

- "dmdaemon Object Parameters" on page 290
- "drivegroup Object Parameters" on page 397
- "filesystem Object Parameters" on page 351
- "libraryserver Object Parameters" on page 393
- "msp Object Parameters for a Cloud MSP" on page 449
- "msp Object Parameters for a DCM MSP" on page 460
- "msp Object Parameters for a Disk MSP" on page 467
- "msp Object Parameters for an FTP MSP" on page 455
- "services Object Parameters" on page 300
- "taskgroup Object Parameters" on page 312
- "volumegroup Object Parameters" on page 411

third-party backups

Backups scheduled through a third-party application, rather than xfsdump. See also *disk backups*, *integrated backups*, and *tape backups*.

THRESHOLD

The percentage of active data on a volume. DMF will consider a volume to be sparse when it has less than this percentage of data that is still active. See "taskgroup Object Parameters" on page 312.

TIMEOUT_FLUSH

The number of minutes after which the VG will flush files to tape. See "volumegroup Object Parameters" on page 411.

TMF_TMMNT_OPTIONS

Command options that should be added to the tmmnt command when mounting a tape. See:

- "device Object Parameters" on page 348
- "drivegroup Object Parameters" on page 397

TMP_DIR

The base pathname for DMF directories in which DMF puts temporary files such as pipes. See "base Object Parameters" on page 278.

TSREPORT_OPTIONS

Additional options that the run_daily_tsreport.sh script will add to the end of the tsreport command line. See "taskgroup Object Parameters" on page 312.

TYPE

The required name for the object. See:

- "`allocationgroup` Object Parameters" on page 435
- "`base` Object Parameters" on page 278
- "`device` Object Parameters" on page 348
- "`dmdaemon` Object Parameters" on page 290
- "`drivegroup` Object Parameters" on page 397
- "`dumpgroup` Object Parameters" on page 343
- "`fastmountcache` Object Parameters" on page 390
- "`filesystem` Object Parameters" on page 351
- "`libraryserver` Object Parameters" on page 393
- "`migrategroup` Object Parameters" on page 427
- "`msp` Object Parameters for a Cloud MSP" on page 449
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Parameters for a Disk MSP" on page 467
- "`msp` Object Parameters for an FTP MSP" on page 455
- "`node` Object Parameters" on page 294
- "`policy` Object" on page 364
- "`resourcescheduler` Object Parameters" on page 432
- "`resourcewatcher` Object Parameters" on page 434
- "`services` Object Parameters" on page 300
- "`taskgroup` Object Parameters" on page 312
- "`volumegroup` Object Parameters" on page 411

unmigratable file

A file that the daemon will never select as a migration candidate.

unmanaged archive filesystem

See *archive filesystem*.

unmigrate

See *recall*.

UPDATE_CTIME

The parameter that specifies whether or not the `st_ctime` value of files in this filesystem should be changed when certain operations are performed. See "`filesystem` Object Parameters" on page 351.

USE_AS_DUMP_DEST

Determines if this filesystem can be used for storing integrated-backup dumps. See "`filesystem` Object Parameters" on page 351/

USE_UNIFIED_BUFFER

Determines how DMF manages its buffers when recalling files on this filesystem. See "`filesystem` Object Parameters" on page 351.

VALID_ROOT_HOSTS

Hosts where the `root` user can perform certain functions similar to the `root` user on the DMF server. See "`base` Object Parameters" on page 278.

VERIFY_POSITION

A parameter that specifies whether the LS write child should (prior to writing) verify that the volume is correctly positioned and that the volume was properly terminated by the last use. See "`drivegroup` Object Parameters" on page 397.

VG

See *volume group*.

VG database entry

The daemon database entry for a file that contains the path or key that is used to inform a particular VG where to locate the copy of the file's data.

VOL_MSG_TIME

Specifies, in seconds, the minimum interval between operator notifications for low–volume and no-volume conditions. See:

- "`allocationgroup` Object Parameters" on page 435
- "`volumegroup` Object Parameters" on page 411

volume

In DMF, a logical area of physical tape, virtual tape, or disk such as COPAN MAID that is used for migrating data.

volume group

A volume group is a of component of an LS that is responsible for managing pool of volumes capable of storing single copies of files. Multiple copies of the same files require the use of multiple VGs. See also *LS*.

VOLUME_GROUPS

The VGs containing volumes that can be mounted on any of the drives within this DG. See "`drivegroup` Object Parameters" on page 397.

voided BFID-set state

A BFID set state that consists of one or more soft-deleted daemon database entries, either incomplete or complete. There is no file. See also *bit-file identifier* and *soft-deleted database entry*.

voiding the BFID

The process of removing the BFID from the file inode and soft-deleting all associated database entries. See also *bit-file identifier* and *soft-deleted database entry*.

VOL record

An entry in the volume (VOL) table of the LS database that contains information about a volume. There is one VOL record for each volume. See also *CAT record, VOL table.*

VOL table

A table in the LS database that contains VOL records. See also *CAT table, VOL record.*

`volumegroup`

The configuration object that defines parameters relating to a pool of volumes mountable on the drives of a specific DG that are capable of holding, at most, one copy of files. See "`volumegroup` Object Parameters" on page 411.

volume-based media

Physical tape, COPAN MAID, or COPAN VTL

`VOLUME_LIMIT`

The maximum number of volumes that can be selected for merging at one time. See "`taskgroup` Object Parameters" on page 312.

volume merging

The mechanism provided by the LS for copying active data from volumes that contain largely obsolete data to volumes that contain mostly active data.

volume serial number

A label that uniquely identifies a specific volume.

VSN

See *volume serial number*

VTL

Virtual tape library.

`WATCHER`

The resource watcher that the LS should run. See "`libraryserver` Object Parameters" on page 393.

`WEIGHT`

The parameter that assigns a weighting to one or more VGs. See "`resourcescheduler` Object Parameters" on page 432.

`WRITE_CHECKSUM`

The parameter that specifies that a block should be checksummed before writing. See:

- "`drivegroup` Object Parameters" on page 397
- "`msp` Object Parameters for a Cloud MSP" on page 449
- "`msp` Object Parameters for a DCM MSP" on page 460
- "`msp` Object Parameters for a Disk MSP" on page 467
- "`msp` Object Parameters for an FTP MSP" on page 455

write child

A data-mover process that migrates data to secondary storage.

zone

A logical grouping of chunks. Zones are separated by file marks. The target size of a zone is configurable by media type.

`ZONE_SIZE`

The parameter that specifies about how much data the write child should put in a zone. See "`volumegroup` Object Parameters" on page 411.

Index

?? ??

1PB+ license, 68
8–character barcodes, 119
10TB+ license, 68
100TB+ license, 68
256b-byte inodes, 104

A

About panel in DMF Manager, 194
absolute block positioning, 42
accelerated access to first byte, 6
active parallel data-mover node, 74
Activity panel in DMF Manager, 193
ADMDIR_IN_ROOTFS, 102, 279
Admin Guide panel in DMF Manager, 194
Admin mode functionality, 196
admin state and fencing, 178
ADMIN_EMAIL, 280
$ADMINDIR, 329
administrative directories, 44
administrative directory permission requirements, 45
administrative tasks
  automated maintenance tasks, 175
  best practices, 123
  daemon configuration, 304
  filesystem backups, 54, 309
  maintenance and recovery, 585
  overview, 52, 53
  tape management, 443
age expression, 381
AGE_WEIGHT, 147, 372, 381, 383, 510, 750
AGGRESSIVE_HVFY, 397
alert records
  remove, 306
ALERT_RETENTION, 118, 312
alerts database size, 117
Alerts panel in DMF Manager, 193, 222
ALGORITHM, 433
allocation group, 41
ALLOCATION_GROUP, 411, 423
ALLOCATION_MAXIMUM, 412
ALLOCATION_MINIMUM, 412
allocationgroup
  allocationgroup object overview, 274
Amazon S3
  See "cloud MSP", 569
AMPEX DIS/DST, 276
API commands, 130
application support, 8
architecture, 38
architecture requirements, 48
archive file requests, 712
archive filesystem and archiving, 119
archives for DMF Manager monitoring, 130
archiving, 5
  overview, 13
  See "DMF direct archiving", 119
archiving files, 15
ASG-Time Navigator, 769
attr, 171
attr2, 171
autolog file, 507
autolog log file, 513
automated maintenance tasks
  daemon configuration, 304

overview, 175
automated space management
administration duties, 53
candidate list generation, 510
commands overview, 62
file exclusion, 510
log, 507
log file, 513
parameters, 369, 376
relationship of targets, 512
selection of migration candidates, 511
automated space management procedure, 385
automatic monitoring, 2
automatic start after reboot, 182
automounters, 42
averages, 242
aware backup packages, 767

B

backup package configuration, 767
backup packages, 767
backups
consistent completion times, 147
COPAN MAID reserved space, 145
databases, 137
DMF configuration file, 102
dump scripts, 141
filesystems, 137
incremental, 148
integrated-backup method, 149
migrate files first, 146
of daemon database, 339
optimizing, 145
other methods, 161
physical tapes, 144
requirement for, 141
run_full_dump.sh, 142
run_partial_dump.sh, 143
SGI 400 VTL tapes, 144
site-specific factors, 144
VTL tapes, 144
balance data among libraries, 139
bandwidth and socket merges, 128
BANDWIDTH_MULTIPLIER, 398
barcodes, 119
base data-capacity license, 67
base metrics, 242
base object
icon in DMF Manager, 213
overview, 272
parameters, 278
basic DMF, 33
basic DMF product, 33
best practices
administrative, 123
configuration, 87
installation, upgrade, and downgrade, 81
LTFS, 632
bfid, 524
bit-file identifier (BFID), 38
black clock symbol, 669
BLOCK_SIZE, 398
blocks, 534
blocksize, 679
blocksize keyword, 558
BOF/bof, 383
borrow a DMF drive for LTFS use, 631
BORROW_FROM_DG, 634
bottlenecks, 134
BUFFERED_IO_SIZE, 352, 461
burst_size, 134
byte range requests and partial-state files, 780

C

CACHE_AGE_WEIGHT, 378, 381
CACHE_DIR, 46, 102, 163, 393, 486
CACHE_MEMBERS, 391
CACHE_SPACE, 106, 394

CACHE_SPACE_WEIGHT, 379, 381
CANCEL message
  FTP MSP, 576
canceling a request, 613
cancelling changes, 219
candidate list
  creation, 509
  generation, 510
  terminology, 53
capability license, 67
capacity
  determination, 72
  DMF software, 42
  license, 67
  overhead and, 42
case study on zone size, 781
CAT records
  backup, 646
  definition, 37
  dmatls database and , 532
  messages, 677
  records and LS database directories, 536
cflags, 548
change notification suppression, 133
chattr, 140
checkage, 520
CHECKSUM_TYPE, 412
checktime, 521, 524
child-process constraints, 122
CHILD_MAXIMUM, 122, 449, 456, 461, 468
chkconfig, 165, 181–183
chkconfig for dmfsoap, 601
chunk table, 622, 624
chunkdata , 548
chunklength, 548
chunknumber, 548
chunkoffset, 548
chunkpos, 548
chunks, 534
chunksleft, 679
chunksleft keyword, 558
CIFS, 9
client port specification, 188
clients
  commands, 57
  installation, 168
  OS supported, 12
cloud MSP
  log files, 571
  msp object for, 449
  request processing, 569
  terminology, 16
CLOUD_BACKEND, 450
CLOUD_BASE_PATH, 450
CLOUD_BUCKET, 450
CLOUD_HOST, 450
CLOUD_PASSWORD, 151, 451
CLOUD_REDUNDANCY, 451
collecting information for problem analysis, 126
COMMAND, 394, 452, 456, 461, 468
commands, 56, 59
commands that are undocumented, 130
commands to run on a copy of the DMF
    database, 125
comments and DMF Manager configuration, 211
communication, 91
COMPRESSION_TYPE, 398
configuration
  allocationgroup object, 435
  automated space management, 369, 376
  backup of, 102
  base object, 278
  best practices, 87
  cloud MSP, 449
  command overview, 58
  considerations, 169
  daemon object configuration, 290
  DCM, 460
  DCM MSP, 464
  device object, 348
  disk MSP, 467
  DMF Manager and, 210

dmmaint and, 791
drivegroup object, 397
dump_tasks, 309
fastmountcache object, 390
file weighting, 371, 378, 386
filesystem object, 351
FTP MSP, 455
initial, 791
libraryserver object, 393
LS objects, 392
LS setup, 444
migrategroup object, 427
msp object
cloud MSP, 449
DCM, 460
DCM MSP, 464
disk MSP, 467
FTP MSP, 455
MSP/VG selection, 374, 380, 388
MSP/VG selection for integrated backups, 389
node object, 294
objects, 58, 272
OpenVault, 502
overview, 165
parameters, 58, 472
See also "parameters", 278
policy object, 364
resourcescheduler object, 432
resourcewatcher object, 434
services object, 300
space management parameters, 511
SPOOL_DIR, 525
stanza, 274
stanza format, 274
taskgroup object, 442
verifying, 487
volumegroup object, 411
configuration file samples, 101
Configuration menu in DMF Manager, 193
configuration pending message, 170
Configuration tab in DMF Manager, 193
Configure button, 790
context manipulation subroutines, 705
converting data from other HSMs, 189
COPAN
configuration best practices, 109
COPAN MAID
fast-mount cache and, 112
COPAN MAID VSN, 64
COPAN RAID set, 8
COPAN VTL stopping, 595
COPAN_VSNS, 102, 394
copy file requests, 710
count directive, 518, 545, 555
cpio file recall, 643
create directive, 518, 545, 555
ctime, 358
current metrics, 242
custom I/O performance charts, 248
customizable policies
See "site-defined policies", 188
customizing DMF, 186
CXFS
basic DMF figure, 33
DMF and, 104
parallel data-mover nodes and, 35
RECALL_NOTIFICATION_RATE and, 107
support for, 8
cxfs_admin, 486
cxfs_recovery_timeout_stalled, 119

D

daemon
commands overview, 59
configuration parameters, 290
configuring automated maintenance tasks, 304
dmd_db.dbd, 647
log, 507
logs and journals, 525
object

See "dmdaemon object", 272
processing, 515
shutdown, 516
startup, 515
tasks, 304
data integrity, 38
administrative tasks and, 54
copying filesystem data, 309
overview, 38
data reduction process and DMF Manager, 130
data reliability
administrative tasks and, 54
copying daemon database, 339
copying filesystem data, 309
data-capability license, 67
data-mover process, 33
DATA_LIMIT, 307, 313, 443
database daemon, 37
database journal files, 527
database loading and journaling, 128
database lock manager incompatibility, 788
DATABASE_COPIES, 305, 313, 339
databases, 37
audit, 305
automated verification task, 337
automating copying for reliability, 339
back up, 305
backup, 646
configuring automated tasks, 339
directory location, 517
dmcatadm, 677
dmdadm and, 517
dmvoladm message, 679
record length, 173
recovery, 649
See "daemon database", 173
selection, 646
size of databases, 97
dataleft, 679
dataleft keyword, 558
datalimit, 560
datawritten keyword, 558
DB_SERVER_BIN, 280
DB_SERVER_HOST, 280
DB_SERVER_PORT, 280
dbrec.dat file, 647
dbrec.keys file, 647
DCM
administration, 307
configuration, 460
DCM disk caches, 246
DCM MSP
child-process constraints, 122
commands, 63
configuration, 464
disk MSP and, 579
filesystems and, 161
terminology, 16
DCM STORE_DIRECTORY rules, 367
DCP disabling, 590
dd, 135
delay icon on Windows systems, 127
delay in accessing files, 668
delete directive, 518, 545, 555
deleteage, 521
deletetime, 521, 524
delimiter in configuration file, 275
device block-size defaults and bandwidth, 276
device object
overview, 273
parameters, 348
DHCP and YAST, 83
direct archiving
See "DMF direct archiving", 13, 119
DIRECT_IO_MAXIMUM_SIZE, 280
DIRECT_IO_SIZE, 352, 461
directory structure prior to DMF 2.8, 783
dirsync and STORE_DIRECTORY, 46
disable components
COPAN VTL, 595
DCP, 590
library, 592

OpenVault drive, 591
TMF drive, 594
disaster recovery
DMF Manager and, 651
DR bundle, 150
DR database, 151
HA and, 657
integrated backups and, 651
overview, 89, 650
Parallel DMF and, 655
See also "DR bundle", 154
DISCONNECT_TIMEOUT, 394
disk cache manager
See "DCM MSP", 579
disk MSP
command, 64
configuration, 467
local, 121
log files, 578
overview, 577
request processing, 577
terminology, 16
verification, 579
disk space capacity, 7
DISPLAY environment variable, 789
DLT, 276
DmaConfigStanzaExists(), 752
DmaContext_t, 739
DmaFrom_t, 740
DmaGetConfigBool(), 753
DmaGetConfigFloat(), 754
DmaGetConfigInt(), 755
DmaGetConfigList(), 756
DmaGetConfigStanza(), 757
DmaGetConfigString(), 758
DmaGetContextFlags(), 759
DmaGetCookie(), 759
DmaGetDaemonMigGroups, 760
DmaGetDaemonVolAndMigGroups(), 760
DmaGetDaemonVolGroups(), 761
DmaGetMigGroupMembers, 761
DmaGetProgramIdentity(), 762
DmaGetUserIdentity(), 762
DmaIdentity_t, 740
DmaLogLevel_t, 742
dmanytag, 785
DMAPI
automatically enabled, 105
kernel interface, 39
requirement, 48
DMAPI on SLES 10, 660
DMAPI_PROBE, 670
dmarchive, 13, 50, 57, 119, 351, 708
dmarchive.php, 601
DmaRealm_t, 742
DmaRecallType_t, 743
DmaSendLogFmtMessage(), 763
DmaSendUserFmtMessage(), 764
DmaSetCookie(), 765
dmatls
journal files, 538
library server terminology, 16
log files, 539
LS operations, 532
VOL records, 537
dmatrc, 42, 532
dmatread, 63, 533, 567
dmatsnf, 63, 533, 568
dmattr, 57
dmattr.php, 601
dmatvfy, 63
dmatwc, 42, 532
dmaudit, 51
changes in DMF 3.2, 787
summary, 60
verifymsp, 568
dmcancel, 263, 606
dmcapacity, 57
dmcatadm
directives, 545
example of list directive, 552
field keywords, 548

interface, 544
keywords, 548
limit keywords, 550
summary, 63
text field order, 553
dmcheck, 60, 487, 658
dmcleardcmtag, 785
dmclearpartial, 786
dmcleartag, 785
dmcloudmsp, 16, 452
dmclripc, 64
dmcollect, 64, 126, 675
dmconfig, 59
dmcopan, 64
dmcopy, 57
dmd_db journal file, 525
dmd_db.dbd, 647
dmdadm
directives, 517, 518
example of list directive, 523
field keywords, 520
format keyword, 522
format keywords, 520
limit keywords, 522
selection expression, 519
summary, 60
text field order, 524
dmdadm –j, 128
dmdaemon (use service command), 124
dmdaemon object
associated task scripts, 305
icon in DMF Manager, 213
overview, 272
parameters, 290
dmdate, 64
dmdbcheck, 56, 60, 63, 125
dmdbrecover, 60, 647
dmdidle, 60
dmdlog log, 507
dmdlog log file, 515, 525
dmdskfree, 63
dmdskmsp, 16, 577
dmdskvfy, 64, 579
dmdstat, 60
dmdstop, 60, 516
dmdstop (use service command), 124
dmdu, 57
dmdump
run only on a copy of the DMF database, 125
summary, 64
text field order, 563
dmdumpj, 64
DMF Activity panel in DMF Manager, 193
DMF administrative directories, 44
DMF administrative directory permission requirements, 45
DMF direct archiving
API subroutines, 712
archive file requests, 712
configuration file and, 351
DmuFilesysInfo(), 708
filesystem object and, 351
overview, 119
requirements, 50
SiteArchiveFile() policy subroutine, 743
DMF file properties, 12
DMF I/O panel in DMF Manager, 193
DMF Manager
About panel, 194
access password, 192
accessing the GUI, 192
acknowledge a command, 233
Activity panel, 193, 235
Admin Guide panel, 194
Admin mode functionality, 196
admin password, 198
Alerts panel, 193, 222
archives, 130, 235
Backups tab, 193
browser support, 49
checkpoint a command, 233
configuration file parameter display, 219

Configuration menu, 193
Configuration tab, 193
configuring DMF, 210
  creating a new object, 217
  deleting an object, 218
  exiting configuration mode, 219
  limitations, 211
  new configuration file, 212
  object menu, 212
  saving changes, 218
  show all objects, 211
  templates, 212
  validating changes, 218
copying an object, 215
DCM MSP monitoring, 246
disaster recovery, 262
DMF Activity panel, 193, 234
DMF I/O panel, 193
DMF Manager Tasks panel, 193
DMF Resources panel, 193, 234
drive state, 242
error messages, 666
filesystem monitoring, 239
Getting Started, 201
Getting Started panel, 193
help, 198
Help menu, 194
hold flags, 231
I/O panel, 193
I/O statistics, 248
installing/deleting licenses, 203
introduction, 9
key to symbols, 199
kill a command, 233
library management, 232
Library panel, 193
library usage, 242
license capacity, 205
Licenses panel, 193
login, 198
menu bar, 192
Messages tab, 193
metrics, 130, 235
modifying an object, 217
monitoring performance, 233
node state, 252
OpenVault library is missing, 668
Overview panel, 10, 192, 193
Parameters panel, 193, 220
password to access the GUI, 192
password to make administrative changes, 198
preferences, 201
problem discovery, 221
queue viewing, 263, 605
quick start, 201
Recovery panel, 193
refreshing the view, 202
relationships among DMF components, 10, 227
Reports panel, 193, 225
requirements, 49
Resources panel, 193
resources statistics, 238
resume a command, 233
starting/stopping DMF, 220
starting/stopping the mounting service, 220
statistics, 666
Statistics menu, 234
Statistics tab, 193
Storage tab, 193
tasks, 233
tips for using, 192
troubleshooting, 666
URLs for, 192
user-generated activity, 235
Volumes panel, 193, 229
“what is” help, 201
DMF Manager Tasks panel in DMF Manager, 193
DMF mover service, 487
DMF Resources panel in DMF Manager, 193
DMF SOAP
  See "SOAP", 597
DMF software

archiving overview, 19
cycle, 2
migration, 7
DMF statistics are unavailable, 666
DMF user library
See "user library (libdmfusr.so)", 681
dmf.conf
See ""configuration" and "parameters"", 59
dmf.conf.cloud, 101
dmf.conf.copan_maid, 101
dmf.conf.copan_vtl, 101
dmf.conf.dcm, 101
dmf.conf.dsk, 101
dmf.conf.fmc, 101
dmf.conf.ftp, 101
dmf.conf.jbfs, 101
dmf.conf.ls, 101
dmf.conf.parallel, 102
dmf_client_ports, 188
dmfdaemon, 61, 515
dmfill, 65, 587
dmfind, 57
dmflicense, 58, 77
dmfsfree, 62, 509
dmfsmon, 62, 369, 376, 509–511
dmfsoap, 600
dmfsoap stop, 601
dmftpmsp, 16, 449, 455, 573
dmfusr.so, 785
dmget, 57
dmget.php, 601
dmhdelete, 61
dmi, 105, 660
dmi mount option, 105
DMIG, 39
dmlocklog log, 507
dmlockmgr
abort, 529
communication and log files, 527
continuous execution, 527
database journal files, 527
interprocess communication, 528
log, 507
overview, 65
transaction log files, 527, 529
dmls, 57
dmmaint
configuration file definition, 791
Configure button, 790
GUI, 789
Inspect button, 790
License Info button, 791
multiple active versions of DMF, 783
overview, 789
Release Note button, 790
Update License button, 791
dmmigrate
file backup, 643
summary, 61
dmmigrate periodic task, 116
DMMIGRATE_MIN_OFFLINE_AGE, 313
DMMIGRATE_MIN_OFFLINE_SIZE, 314
DMMIGRATE_MINIMUM_AGE, 117, 313
DMMIGRATE_QUEUE_MAX, 314
DMMIGRATE_TRICKLE, 117, 314
DMMIGRATE_UNACK, 314
DMMIGRATE_VERBOSE, 117, 314
DMMIGRATE_WAIT, 117, 314
dmmove, 65, 136, 292, 579, 580, 642
dmmvtree, 65
dmn_connect table, 622, 625
dmn_request table, 622, 626
dmnode_admin, 488
dmoper, 57, 185
dmoper.php, 602
dmov_keyfile, 65, 498
dmov_loadtapes, 65, 505
dmov_makecarts, 65, 504
dmprojid, 65
dmprojid.php, 602
dmput, 57
dmput.php, 602

dmqview, 66, 605
dmrepri, 64, 66, 263, 606
dmrestore, 66
dmrestore.php, 602
dmscanfs, 62, 306, 511
dmselect, 66, 580
dmsnap, 61
dmsort, 66
dmstat, 66
dmtag, 57, 358, 785
dmtag.php, 602
dmtapestat, 66
DMU_DEF_PRIORITY, 701
DMU_MAX_PRIORITY, 701
DMU_MIN_PRIORITY, 701
DmuAllErrors_t, 686
DmuArchiveAsync(), 712
DmuArchiveSync(), 712
DmuAttr_t, 687
DmuAwaitReplies(), 728
DmuByteRange_t, 688
DmuByteRanges_t, 689
DmuChangedDirectory(), 707
DmuCompletion_t, 692
DmuCopyAsync_2(), 710
DmuCopyRange_t, 693
DmuCopyRanges_t, 693
DmuCopySync_2(), 710
DmuCreateContext(), 705
DmuDestroyContext(), 707
DmuErrHandler_f, 694
DmuErrInfo_t, 695
DmuError_t, 695
DmuEvents_t, 696
DmuFhandle_t, 696
DmuFilesysInfo(), 708
DmuFsysInfo_t, 696
DmuFullRegbuf_t, 697
DmuFullRegion_t, 698
DmuFullstat_2_t, 699
DmuFullstat_t, 698
DmuFullstatByFhandleAsync(), 714
DmuFullstatByFhandleSync(), 714
DmuFullstatByPathAsync(), 714
DmuFullstatByPathSync(), 714
DmuFullstatCompletion(), 729
DmuGetByFhandleAsync_2(), 720
DmuGetByFhandleSync_2(), 720
DmuGetByPathAsync_2(), 720
DmuGetByPathSync_2(), 720
DmuGetNextReply(), 730
DmuGetThisReply(), 732
dmunput, 66
DmuOperandType_t, 700
DmuPriority_t, 701
DmuPutByFhandleAsync(), 717, 748
DmuPutByFhandleSync(), 717, 748
DmuPutByPathAsync(), 717, 748
DmuPutByPathSync(), 717, 748
DmuRegion_t, 701
DmuRegionbuf_t, 701
DmuReplyOrder_t, 702
DmuReplyType_t, 702
dmusage, 58, 72
DmuSettagByFhandleAsync(), 725
DmuSettagByFhandleSync(), 725
DmuSettagByPathAsync(), 725
DmuSettagByPathSync(), 725
DmuSeverity_t, 702
DmuVolGroup_t, 703
DmuVolGroups_t, 703
dmversion, 58
dmversion.php, 602
dmvoladm
  directives, 555
  examples of list directive, 564
  field keywords, 558
  format keywords, 561
  limit keywords, 560
  select directive, 543
  summary, 63
  text field order, 563

VOL records and, 538
dmxfsrestore, 66
DNS, 92, 93
do_predump.sh
snapshot location, 768
summary, 768
Time Navigator, 770
domain name service, 93
downgrade
best practices, 81, 87
partial-state file feature and, 786
DR bundle, 154
See "disaster recovery", 150
DR database
See "disaster recovery", 151
drive disabling, 591
drive does not exist, 661
drive entry error, 660
drive group
object, 273
OpenVault and, 502
terminology, 40
TMF tapes and, 505
drive visibility, 49
drive-group names, 121
DRIVE_GROUPS, 102, 116, 394
DRIVE_MAXIMUM, 99, 399, 413
DRIVE_SCHEDULER, 400
drivegroup, 116
drivegroup object
overview, 273
parameters, 397
drives
performance improvements, 105
zone size and, 105
DRIVES_TO_DOWN, 400
DRIVETAB, 315
DSK_BUFSIZE, 461, 468
DSO, 41
dual-residence, 367
dual-resident state, 579
dual-state file, 19, 28
file migration and, 3
terminology, 15
xfsdump and, 643
DUALRESIDENCE_TARGET, 377
dump directive, 518, 545, 555
dump utilities, 55
DUMP_COMPRESS, 315, 343, 352
DUMP_CONCURRENCY, 316
DUMP_DATABASE_COPY, 768, 769
DUMP_DESTINATION, 150, 316
DUMP_DEVICE, 317
DUMP_DR_EMAIL, 317
DUMP_DR_EMAIL_MAX_SIZE, 317
DUMP_DR_MINIMAL, 317
DUMP_DR_MIRRORS, 317
DUMP_FILE_SYSTEMS, 318, 344, 768
DUMP_FLUSH_DCM_FIRST, 163, 319, 462, 768
DUMP_GROUPS, 319
DUMP_INVENTORY_COPY, 319
DUMP_MAX_FILESPACE, 148, 320, 344, 354
DUMP_MIGRATE_FIRST, 137, 146, 163, 320, 345, 354, 768
DUMP_MIRRORS, 177, 321
DUMP_MISC_LOCATIONS, 151, 321
DUMP_POST_BACKUP_SCRIPT, 321
DUMP_POST_DR_SCRIPT, 322
DUMP_PRE_BACKUP_SCRIPT, 322
DUMP_RETENTION, 143
ASG-Time Navigator, 770
run_hard_deletes.sh, 306
summary, 323
DUMP_STREAMS
summary, 323, 345, 354
DUMP_TAPES, 186, 324
dump_tasks, 309
DUMP_VSNS_USED, 324
DUMP_WORKSPACE
summary, 324
DUMP_XFSDUMP_PARAMS, 324, 345, 354
dumps.db, 149

Dynamic Shared Object library, 41

E

empty damaged volume in DMF Manager, 232
empty graphs, 668
end of life
  tape autoloader API, 783
  tape MSP, 784
enhanced-NFS RPC corruption, 131
entitlement ID, 74
entries keyword, 551
EOF, 384
EOT error, 660
eotblockid keyword, 558
eotchunk, 679
eotchunk keyword, 558
eotpos, 679
eotpos keyword, 559
eotzone, 680
eotzone keyword, 559
error messages in DMF Manager, 193
error reports and tapes, 443
/etc/dmf/dmbase, 784
/etc/dmf/dmf.conf, 472, 667
/etc/hosts, 91
/etc/hosts file, 92
/etc/lk/keys.dat, 75
/etc/nsswitch.conf, 92
/etc/resolv.conf, 93
/etc/sgi-ltfs.conf, 633
/etc/tmf/tmf.config, 126
/etc/xinetd.conf, 108
/etc/xinetd.d/auth, 166, 167
/etc/xinetd.d/pidentd, 166, 167
/etc/xinetd.d/tcpmux, 108
examples
  /etc/hosts file, 92
  /etc/inet/hosts file, 92
  name services, 92
explicit start, 183
explicit start dmfsoap, 601
explicit stop, 184
EXPORT_METRICS, 102, 280, 667
EXPORT_QUEUE, 290
extended attribute structure, 171
extension records, 104

F

fabric, 49
FADV_SIZE_MAID, 400
FADV_SIZE_MSP, 462, 468
fast-mount cache
  configuration best practices, 112
  definition, 16
  diagrams, 24
  merging and, 113
  multiple migration copies and, 100
  overview, 30
  requirements, 50
fastmountcache, 112
fastmountcache object
  overview, 273
  parameters, 390
feature history, 783
file concepts, 14
file hard deletion, 306
file migration
  See "migration", 3, 510
file ranking, 53
file recall, 28
file regions, 5
file request subroutines, 709
file tagging, 186
file weighting, 365, 371, 378, 386
FILE_RETENTION_DAYS, 114, 307, 308, 325
filesize keyword, 549
filesys table, 622, 627
filesystem errors, 658

filesystem information subroutine, 708
filesystem object
  overview, 273
  parameters, 351
filesystems
  back up, 306
  conversion, 462, 469
  DCM MSP and, 161
  dmdskmsp, 462, 469
  dmftpmsp, 452, 457
  migrate, 305, 307
  mount options, 83
  report on, 306
  scan, 306
filters, 225
FINISH message
  FTP MSP, 576
Firefox and DMF Manager, 10, 49
flag keywords, 561
FLUSHALL message
  FTP MSP, 576
FMC
  See "fast-mount cache", 50
FMC_MOVEFS, 114, 307, 308, 325
FMC_NAME, 325
force migration, 107
format keyword, 523, 551
FORWARD_RECALLS, 140, 185, 413
free space management, 7, 53
free-space minimum threshold, 2, 7
FREE_DUALRESIDENT_FIRST, 377
FREE_DUALSTATE_FIRST, 369
FREE_SPACE_DECREMENT, 369, 377, 512
FREE_SPACE_MINIMUM, 369, 377, 511
FREE_SPACE_TARGET, 370, 378
FREE_VOLUME_MINIMUM, 113, 307, 308, 325
FREE_VOLUME_TARGET, 113, 307, 308, 326
FTP, 8
FTP MSP
  log files, 575
  messages, 576
  msp object for, 455
  overview, 573
  request processing, 573
  terminology, 16
FTP_ACCOUNT, 456
FTP_COMMAND, 456
FTP_DIRECTORY, 456
FTP_HOST, 456
FTP_PASSWORD, 151, 457
FTP_PORT, 457
FTP_USER, 457
FULL_THRESHOLD_BYTES, 468
FULL_ZONE, 414
fullstat requests, 714

G

get file requests, 720
GET_WAIT_TIME, 140, 414
Getting Started panel in DMF Manager, 193
gid expression, 381
globals table, 622
gmgrd, 668
gray background in DMF Manager, 213
GROUP_MEMBERS, 102, 428
GUARANTEED_DELETES, 122, 452, 457, 462, 469
GUARANTEED_GETS, 122, 452, 457, 462, 469
GUI
  See "DMF Manager", 9

H

h1, 562
HA
  differences in administration and configuration, 125
  DMF support, 13
  license requirements, 68
HA cluster and DNS, 93

hard-deleted files
maintenance/recovery, 586
run_hard_deletes.sh task, 306
hardware requirements, 43
HBA drivers, 107
HBA_BANDWIDTH, 280, 295
he, 561
help directive, 518, 545, 555
Help menu in DMF Manager, 194
helper subroutines for sitelib.so, 752
herr, 561
hexadecimal number, 384
hextern, 186, 561
hf, 559
hflags, 559, 562
hfree, 562
HFREE_TIME, 415
hfull, 562
hierarchical storage management, 5
high availability
See "HA", 13
historical feature information, 783
hl, 562
hlock, 562
ho, 562
hoa, 562
hold flags, 231, 558
HOME_DIR, 45, 97, 281, 486, 533
host port speeds and tape drives, 134
hostname mapping, 91
hosts file, 92
HP ULTRIUM, 276
hr, 562
hro, 562
hs, 562
hsite*, 562
HSM conversion to DMF, 189
HSM data import, 189
hsparse, 562
HTML_REFRESH, 434
hu, 562
hv, 562
HVD disk, 8
hvfy, 562
hx, 561

I

I/O panel in DMF Manager, 193
IBM 03590, 276
IBM TS1140, 276
IBM ULT3580, 276
IBM ULTRIUM, 276
ident service, 47, 166
immutable bit, 140
IMMUTABLE_BFIDS, 354
IMPORT_DELETE, 457, 469
IMPORT_ONLY, 415, 457, 469
importing data from other HSMs, 189
incremental backups, 148
incremental data-capacity license, 67
initial configuration, 791
initial planning, 52
initrd, 107
INITRD_MODULES, 107
inode, 15
inode size, 171
inode-resident extended attributes, 104
inodes, 5
Inspect button, 790
inst, 183, 184, 601
installation, 170
best practices, 81
client installers on DMF server, 169
considerations, 169
ISSP release, 169
overview, 165
procedure, 165
installation source, 52
instances parameter, 108
integrated backups

pruning files, 142
See "backups", 141
integrated data-mover functionality, 35
INTERFACE, 121, 295
Internet Explorer and DMF Manager, 10, 49
interoperability, 8
interprocess communication (IPC), 175, 527, 528
introduction, 1
IOStreamGuard, 179
IPv4, 91, 93
IPv6, 91, 93
IRIX
client platform, 12
IRIX conversion, 773

J

JBFS, 8
joining of byte ranges, 385
journal files
configuring automated task for retaining, 338
database, 527
dmfdaemon, 525
LS, 538
remove, 306
retaining, 586
summary, 56
JOURNAL_DIR, 45, 98, 163, 281, 525, 533, 538
JOURNAL_RETENTION, 306, 326, 526, 539
JOURNAL_SIZE, 282, 526, 538, 539
journaling and database loading, 128

K

keys.dat, 75
KILL_WC_IF_DEMAND, 412
Knowledgebase, 675

L

label keyword, 559
LABEL_TYPE, 400
LCP and COPAN, 110
libdmfadm.H, 739
libdmfcom.H, 739
libdmfusr.so, 58, 187
See "user library (libdmfusr.so)", 681
libraries, 242
See "site-defined policy library", 735
See "user library (libdmfusr.so)", 681
library disabling, 592
Library panel in DMF Manager, 193
library server
See "LS", 392
library slot usage, 242
library versioning, 684
libraryserver, 116
libraryserver object
associated task scripts, 307
overview, 273
parameters, 393
libsrv_db journal file, 538
libsrv_db.dbd, 537, 538, 647
license capacity, 205
License Info button, 791
License Keys (LK), 67
LICENSE_FILE, 102, 282
Licenses panel in DMF Manager, 193
licensing, 67
capability license, 67
capacity determination, 72
commands, 58
data-capacity license, 67
dmflicense, 58, 77
dmusage, 58
entitlement ID, 74
/etc/lk/keys.dat, 282
file, 282
HA and, 68

host information, 74
installation, 75
keys, 75
License Keys (LK), 67
LICENSE_FILE, 282
lk_hostid, 74
lk_verify, 77
mounting services and, 74
obtaining from SGI, 75
OpenVault and, 74
Parallel Data-Mover Option and, 74
requirements, 48
SGI webpage, 79
stored capacity and, 67
TMF and, 74
types, 67
verification, 75
lights-out operations, 52
limit keywords
dmcatadm, 550
dmvoladm command, 560
Linux
partial-state files and, 779
list directive, 518, 545, 555
LK license, 67
lk_hostid, 74
lk_verify, 77
load directive, 518, 545, 555
LOCAL_, 125
lock manager, 527
log files
automated space management, 513
automated task for retaining, 338–340
changes in DMF 3.2, 787
cloud MSP, 571
disk MSP, 578
dmfdaemon, 525
dmlockmgr communication and, 527
FTP MSP, 575
LS, 539
remove, 306
retaining, 585
scan for errors, 307
transaction log files, 529
LOG_RETENTION, 306, 308, 326
LOGICAL_BLOCK_PROTECTION, 416
login for DMF Manager, 198
logs
general format, 507
low-voltage differential (LVD) tapes, 8
LS
architecture, 39
CAT records, 532, 536
commands, 57
configuration example, 436
database, 538
database recovery, 649
directories, 533
dmatread, 567
dmatsnf, 568
dmaudit verifymsp, 568
dmcatadm, 544
dmvoladm, 554
drive scheduling, 582
error analysis and avoidance, 580
journals, 538
log files, 539
objects, 273, 393
operations, 532
process, 40
setup, 392
status monitoring, 582
tape operations, 532
tape setup, 444
terminology, 15
VOL records, 533, 537
volume merging, 543
LS commands, 63
LS database, 37
LS_NAMES, 102, 116, 290
LSI FC ports and N-port technology, 133
lsiutil, 134

LTFS
  best practices, 632
  configuration, 633
  features, 631
  requirements, 632
LTFS and DMF Manager, 263
LTFS_CART_GROUP, 634
LTFS_DRIVE_GROUP, 634
LTFS_MAX_DRIVES, 635
LTFS_MOUNT_ROOT, 635
LTFS_OV_KEYFILE, 635
LTO-5, 631
LTO-6, 631
Lustre filesystem and DMF archiving, 13, 119
LVD tapes, 8

M

Mac OS X, 12
MAID
  configuration best practices, 109
maintenance and recovery
  automated, 175
  cleaning up journal files, 586
  cleaning up log files, 585
  daemon configuration, 304
  database backup, 646
  databases , 649
  dmfill, 587
  dmmaint, 789
  hard-deletes, 586
  soft-deletes, 586
maintenance task configuration, 442
managed filesystem policy parameters in DMF configuration file, 368
managed filesystem rules, 366
managing the DMF environment
  See "DMF Manager", 9
manypartial, 786
MAX_ALERTDB_SIZE, 118, 326
MAX_CACHE_FILE, 395
MAX_CHUNK_SIZE, 416
MAX_IDLE_PUT_CHILDREN, 417
MAX_MANAGED_REGIONS, 122, 355
MAX_MS_RESTARTS, 170, 401
MAX_PERFDB_SIZE, 118, 306, 327
MAX_PUT_CHILDREN, 99, 401, 417
maximum burst size, 134
media, 52
media concepts, 534
media transports, 42
media-specific processes
  See "MSP", 15
Mediaflux requirements, 51
memory-mapping issues, 131
MERGE_CUTOFF, 418
MERGE_INTERFACE, 121, 295
MERGE_THRESHOLD, 113, 418
merging and fast-mount cache, 113
merging sparse tapes
  run_merge_mgr.sh, 307
  run_tape_merge.sh, 443
merging sparse volumes
  DMF Manager and, 232
  run_merge_stop.sh, 443
MESSAGE_LEVEL, 291, 300, 356, 395, 452, 458, 462, 469
messages
  dmcatadm, 677
  dmvoladm, 679
  FTP MSP, 576
  log, 507
Messages tab in DMF Manager, 193
metrics in DMF Manager monitoring, 130, 235
METRICS_RETENTION, 282
MG
  objects, 273
MiB vs MB and DMF Manager, 234, 236
migrate files via Windows client, 12
migrate group
  configuration best practices, 111

COPAN and, 110
migrated data movement between MSPs, 579
migrated file, 3
migrategroup object
overview, 273
parameters, 428
migrating data from other HSMs, 189
migration
automated file selection, 511
file exclusion, 510
file selection, 511
frequency, 7
MSP/VG, 388, 389
MSP/VG selection, 374, 380
multiple copies of a file, 100
overview, 5, 27
policies, 7
real-time partitions and, 513
recalling, 28
relationship of space management targets, 512
target, 509
terminology, 15
triggers, 7
weighting of files, 371, 378, 386
migration policies, 7
migration targets, 7
MIGRATION_LEVEL, 291, 356, 366, 367, 462
MIGRATION_TARGET, 371
MIN_ARCHIVE_SIZE, 356
MIN_DIRECT_SIZE, 357, 463
MIN_VOLUMES, 419
mkfs.xfs, 136, 171
modification prevention, 140
modifications to the DMF configuration, 102
monitoring DMF, 125
monitoring performance, 233
mount errors, 671
mount options, 83
MOUNT_BLOCKED_TIMEOUT, 402
MOUNT_SERVICE, 348, 402
MOUNT_SERVICE_GROUP, 349, 402
MOUNT_TIMEOUT, 403
mounting service tasks
OpenVault, 491
TMF, 505
mounting services
See “OpenVault” or “TMF”, 13
MOVE_FS, 46, 98, 163, 291, 292, 486
mover table, 623, 628
MSG_DELAY, 403
MSGMAX, 175
MSGMNI, 175
MSP
automated maintenance tasks, 338
CAT records, 536
commands, 57
configuration, 388, 389
disk, 577
dmatread, 567
dmcatadm message, 677
dmfdaemon, 532
dmvoladm message, 679
FTP, 573
log files, 338
log files and automated maintenance tasks, 442
logs, 507
moving migrated data between MSPs, 579
msp object
cloud MSP, 449
DCM, 460
DCM MSP, 464
disk MSP, 467
FTP MSP, 455
overview, 274
selection for migrating files, 374, 380
tape setup, 444
tasks, 443
terminology, 15
types, 17
MSP objects, 446
MSP/VG selection, 366
MSP_NAMES, 102, 290

msp_request table, 623, 629
msp_tasks, 442
mspkey, 521, 524
msplog
  message format, 507
msplog file
  disk MSP, 578
  dmatls, 541
  LS logs, 539
  LS statistics messages, 541
mspname, 521, 524
MULTIPLIER, 102, 428
MULTITAPE_NODES, 404
MVS_UNIT, 458

N

N-port technology, 133
n-tier capability, 16
NAME_FORMAT, 116, 173, 453, 458, 463, 470
Network Information Service, 91
network service configuration and YAST, 83
NFS, 8
nfsd_workaround, 132
NIS, 91, 92
node object
  overview, 272
  parameters, 294
node status in DMF manager, 252
NODE_ANNOUNCE_RATE, 300
NODE_BANDWIDTH, 283, 295
NODE_TIMEOUT, 300
NTP, 124
number of copies, 7

O

objects in DMF configuration file
  allocationgroup object parameters, 435
  base object, 278
  device object, 348
  dmdaemon object, 290
  drivegroup object, 397
  fastmountcache object, 390
  filesystem object, 351
  libraryserver object, 393
  migrategroup object, 427
  msp object
    cloud MSP, 449
    DCM, 460
    DCM MSP, 464
    disk MSP, 467
    FTP MSP, 455
  node object, 294
  overview, 272
  policy object, 364
  resourcescheduler object, 432
  resourcewatcher object parameters, 434
  services object, 300
  stanza format, 274
  taskgroup object, 304, 442
  volumegroup object, 411
offline data management, 54
offline file, 3, 15, 19, 28
online access, 5
only_from, 166, 167
OpenVault, 121
  availability, 170
  considerations, 170
  downgrade from DMF 4.0, 87
  key file, 283
  license, 74
  OV_KEY_FILE, 283
  OV_SERVER, 283
  parameters, 283
  server
    DMF parameter, 283
    parameter for LTFS, 635
  support for, 13
  YAST and, 83

OpenVault configuration tasks
add the dmf application, 494
DMF and OpenVault servers differ, 502
drive groups, 502
initial server configuration, 492
parallel data-mover node configuration, 498
OpenVault drive disabling, 591
OpenVault libraries, 242
OpenVault library disabling, 592
OpenVault server , 122
operations timeout on Windows, 669
operative_flags, 745
operative_priority, 745
operative_volgrps, 745
origage, 521
origdevice, 521, 524
origin file error, 669
originode, 521, 524
origname, 521, 524
origsize, 521, 524
origtime, 521, 524
origuid, 521, 524
orphaned inventory entries, 142
out-of-library tapes, 14
out-ot-library tapes, 185
OV_ACCESS_MODES, 349, 404
ov_admin, 83
ov_dcp, 590
ov_drive, 591, 595
ov_dumptable, 671
OV_INTERCHANGE_MODES, 349, 405
OV_KEY_FILE, 102, 283, 487, 498
ov_library, 185, 592
OV_SERVER, 102
DMF parameter, 283
LTFS parameters, 635
ov_start, 595
ov_stat, 592, 593
overhead of DMF software, 42
oversubscription, 5
Overview panel in DMF Manager, 192, 193

P

Parallel Data Mover Option
private network, 121
parallel data-mover node
requirements, 48
Parallel Data-Mover Option
active node, 74
configuration, 485
CXFS and, 105
disabling / reenabling nodes, 489
installation, 168
license, 67
node state, 488
overview, 33
terminology, 33
Parallel DMF, 6
parameters
DMF configuration parameters, 472
SGI LTFS configuration parameters, 634
Parameters panel in DMF Manager, 193, 220
partial state, 122
partial-state file
CACHE_SPACE_WEIGHT, 379
considerations, 779
enable / disable feature, 292
file regions and, 5
Linux kernel support lacking, 779
online retention, 6
performance cost, 779
recall, 6
SPACE_WEIGHT, 373
terminology, 15
partial-state filed
exact byte range requests, 780
PARTIAL_STATE_FILES, 122, 292
passwords for DMF manager
GUI access, 192
passwords in DMF Manager
admin, 198
path segment extension record, 173

path segment extension records, 104
pathseg, 173
pathseg.dat file, 647
pathseg.keys file, 647
PCP, 667
pcp-storage, 130
PENALTY, 433
PERF_RETENTION, 118, 251, 306, 327
performance archives, 130
Performance Co-Pilot, 130, 235
performance monitoring, 233
performance records, 117
performance statistics
  remove, 306
PERFTRACE_METRICS, 284
periodic maintenance tasks, 304
permission requirements, 45
persistent reservation, 671
pg_data, 47
pg_tmp, 47
pg_xlog, 47
pglogs, 47
php-curl, 602
php5-openssl, 602
php5-soap, 602
pidentd, 166
pipes, 27
pminfo, 667
POLICIES, 357, 366, 463
policies, 186
policy object
  overview, 273
  parameters, 364
poor migration performance, 669
port speeds and tape drives, 134
POSITION_RETRY, 406
POSITIONING, 405
POSIX_FADVISE_SIZE, 357
postgres, 280
postgres user, 47
PostgreSQL
  requirements, 47
  updates, 136
precedence of objects for integrated backups, 154
preconfigured samples in DMF Manager, 212
preferences in DMF Manager, 201
prevent file modification, 140
prevent recalls, 140
preventing automatic start, 181, 183
preventing automatic start of dmfsoap, 601
PRIORITY_PERIOD, 463
private filesystem of DMF and backups, 163
private network, 121
project ID via Windows client, 12
projid, 381
prune-files, 142
prune-inv, 142
prune-only, 143
purge records, 117
put file requests, 717
PUT_IDLE_DELAY, 419
PUTS_TIME, 419

Q

QLogic FC switch, 178
QUANTUM, 276
queue viewing
  canceling a request, 613
  database tables and fields, 620
  dmqview, 605
  modifying report output, 615
  overview, 605
  QView tab, 605
  reports, 606
  reprioritizing, 615
  snapshot, 606
quit directive, 518, 545, 555
QView tab, 263, 605

R

RAID set, 8
range tokens
  RECALL_NOTIFICATION_RATE and, 107
ranges clause, 383
ranking of files, 53
RDM lock manager, 527
READ_ERR_MAXIMUM, 406
READ_ERR_MINIMUM, 407
READ_ERR_TIMEOUT, 407
READ_IDLE_DELAY, 407, 419
READ_TIME, 420
readage, 549
readcount, 549
readdate, 549
recall files via Windows client, 12
recall of migrated files, 28
RECALL_NOTIFICATION_RATE, 107, 292
record length, 173
recordlimit, 522, 550, 561
recordorder, 522, 550, 561
recovery
  databases, 649
Red Hat Enterprise Linux, 48
Red Hat Enterprise Linux (RHEL), 12
REDIRECT_TIME_DG_DOWN, 407
regions, 5
regular file, 15, 28
REINSTATE_DRIVE_DELAY, 408, 581
REINSTATE_VOLUME_DELAY, 408
relationships in DMF Manager, 227
Release Note button, 790
RELEASETHREADS, 358
reliability, 339
REMALERT_PARAMS, 118, 306, 327
remote connection failures, 670
REMPERF_PARAMS, 118, 307, 328
repair directive, 555
replace bad copy with good copy, 642
reporting problems to SGI, 675
Reports Panel in DMF Manager, 225
Reports panel in DMF Manager, 193
reprioritizing a request, 615
request completion subroutines, 727
request processing
  disk MSP, 577
  FTP MSP, 573
requirements
  direct archiving, 50
  DMAPI, 49
  DMF Manager, 49
  DMF SOAP, 50
  fast-mount cache, 50
  hardware, 48
  ksh, 48
  licensing, 48
  mounting service, 48
  OpenVault, 48
  parallel data-mover node, 48
  server-node, 48
  software, 48
  TMP, 48
RESERVED_VOLUMES, 420
resource scheduler
  algorithm, 41
  configuration, 273
  object overview, 273
  object parameters, 432
  resourcescheduler object, 273
  terminology, 41
  weighted_roundrobin, 433
resource watcher
  resourcewatcher object overview, 273
  terminology, 41
Resources panel in DMF Manager, 193
retention of journal files, 338
retention of log files, 338–340
REWIND_DELAY, 408, 421
RHEL, 48
robotic library, 16
root squashing, 154

ROTATION_STRATEGY, 102, 429
ROUND_ROBIN_BY_FILES, 429
rounding of byte ranges, 385
rpm, 183, 184
rpm dmfsoap, 601
RRS, 451
RSCN, 178
rules for policy parameters in DMF configuration file, 366
run_audit.sh, 305, 337
run_compact_tape_report.sh, 852
run_copy_databases.sh, 56, 137, 305, 339
run_daily_drive_report.sh, 305
run_daily_tsreport.sh, 305, 338
run_dcm_admin.sh, 307
run_dmmigrate.sh, 116, 305, 307
run_filesystem_scan.sh, 306, 337
run_fmc_free.sh, 113, 307, 308
run_full_dump.sh, 55, 142, 306, 333, 334
run_hard_deletes.sh, 56, 131, 306, 333, 334
run_merge_mgr.sh, 307
run_merge_stop.sh, 307, 443
run_partial_dump.sh, 55, 143, 148, 306, 333, 334
run_remove_alerts.sh, 117
run_remove_journals.sh, 56, 306, 338, 442
run_remove_logs.sh, 56, 251, 306, 308, 338–340, 442
run_remove_perf.sh, 306
run_scan_logs.sh, 307, 338
run_tape_merge.sh, 307, 442, 443
run_tape_report.sh, 852
run_tape_stop.sh, 442
RUN_TASK, 113, 328, 395, 409, 421

S

S3
  See "cloud MSP", 569
safe modifications to the DMF configuration, 102
safety, 6
SAM, 189
sample configuration files, 101
sample DMF SOAP client files, 601
sample_sitelib.C, 736
sample_sitelib.mk, 736
samples in DMF Manager, 212
SAN switch zoning or separate SAN fabric, 49
scalability, 6
Scality
  See "cloud MSP", 569
SCAN_FILESYSTEMS, 329
SCAN_FOR_DMSTAT, 329
SCAN_OUTPUT, 330
SCAN_PARALLEL, 330
SCAN_PARAMS, 330
script names, 104
SCSI low-voltage differential (LVD) tapes, 8
sdparm, 134
SEAGATE ULTRIUM, 276
secondary storage
  See also "migration", 5
select directive, 555
select system call, 516
SELECT_LOWER_VG, 380, 381
SELECT_MSP, 374, 381, 750
SELECT_VG, 139, 147, 374, 381, 750
selection expression, 556
SELinux, 136
separate SAN fabric, 49
SEQUENTIAL, 429
serial ATA, 16
server capability license, 67
server node functionality, 32
server_args, 166
SERVER_NAME, 284, 488
SERVICES, 296
services object
  associated task scripts, 308
  overview, 272
  parameters, 300
SERVICES_PORT, 102, 301, 488
SessTimeout, 127

set directive, 518, 545, 555
Setprojid file requests, 723
Settag file requests, 725
SGI 400 VTL
  disable drives before stopping, 129
SGI InfiniteStorage Software Platform (ISSP), 166
SGI Knowledgebase, 675
SGI LTFS
  See "LTFS", 632
SGI x86_64 hardware, 48
sgi-dmapi, 105
sgi-ltfs, 631
sgi-ltfs.conf, 633
sgi-xfsprogs, 105
sgi_ltfs, 66
sginfo, 135
shutdown, 183, 184, 529
shutdown of dmfsoap, 601
silo, 16
site tag feature, 785
site tags via Windows client, 12
site-defined policies
  considerations, 738
  customizing DMF, 186
  DmaConfigStanzaExists(), 752
  DmaContext_t, 739
  DmaFrom_t, 740
  DmaGetConfigBool(), 753
  DmaGetConfigFloat(), 754
  DmaGetConfigInt(), 755
  DmaGetConfigList(), 756
  DmaGetConfigStanza(), 757
  DmaGetConfigString(), 758
  DmaGetContextFlags(), 759
  DmaGetCookie(), 759
  DmaGetDaemonMigGroups, 760
  DmaGetDaemonVolAndMigGroups(), 760
  DmaGetDaemonVolGroups(), 761
  DmaGetMigGroupMembers, 761
  DmaGetProgramIdentity(), 762
  DmaGetUserIdentity(), 762
  DmaIdentity_t, 740
  DmaLogLevel_t, 742
  DmaRealm_t, 742
  DmaRecallType_t, 743
  DmaSendLogFmtMessage(), 763
  DmaSendUserFmtMessage(), 764
  DmaSetCookie(), 765
  getting started, 736
  SiteArchiveFile(), 743
  SiteCreateContext(), 745
  SiteDestroyContext(), 746
  SiteFncMap_t, 743
  SiteKernRecall(), 746
  sitelib.so data types, 739, 742
  SitePutFile(), 748
  SiteWhen(), 750
  subroutines overview, 735
  terminology, 188
site-specific configuration parameter and stanza names, 125
site-specific objects and parameters
  DMF Manager and, 211
SITE_SCRIPT, 371, 378
SiteArchiveFile() sitelib.so subroutine, 743
SiteCreateContext() sitelib.so subroutine, 745
SiteDestroyContext() sitelib.so subroutine, 746
sitefn expression, 381
SiteFncMap variable, 737
SiteFncMap_t object, 743
SiteKernRecall() sitelib.so subroutine, 746
SITELIB parameter, 737
sitelib.so
  See "site-defined policy library", 735
SitePutFile() sitelib.so subroutine, 748
sitetag, 358
sitetag expression, 381
SiteWhen() sitelib.so subroutine, 750
size expression, 381
size of DMF database filesystems, 97
SLES, 48
slot usage, 242

small files, 107
small files and DMF, 146
SMB, 9
SMB request timeouts, 127
SMB/CIFS network share, 668
.so file, 41
SOAP, 50
  accessing the GUI, 599
  dmfsoap service, 601
  sample client files, 601
  security/authentication, 601
  See "", 597
  starting, 601
  stopping, 601
  URLs for, 599
SOAP web service, 13
socket merges, 128
soft-deleted files
  maintenance/recovery, 586
softdeleted expression, 381
software mix, 81
software requirements, 43
Solaris, 13
SONY SDX, 276
SONY SDZ, 276
space expression, 381
space management
  commands overview, 62
  DCM MSP and, 513
SPACE_WEIGHT, 147, 373, 381, 383, 510, 750
sparse tapes
  configuration of automated merging, 443
  merging, 307, 443
sparse volumes
  automated merging, 443
  merging, 543
  terminology, 54
special files, 27
SPOOL_DIR, 45, 98, 102, 163, 284, 486, 513, 525, 533
st_ctime, 358
stalled-recovery timeout, 118
start DMF, 124
starting the DMF environment, 181
Statistics tab in DMF Manager, 193
stdin, stdout, stderr and sitelib.so, 738
STK, 276
stop DMF, 124
stopping the DMF environment, 181
Storage tab in DMF Manager, 193
STORE_DIRECTORY, 46, 464, 470, 486, 579
STORE_DIRECTORY and dirsync, 46
SUSE Linux Enterprise Server (SLES), 12, 48
swdn, 134
switch
  QLogic, 178
switch zoning, 49

T

T10000C, 631
T10000D, 631
T10000T2, 631
tape autoloader API end of life, 783
tape barcodes, 119
tape drivers
  ts, 66, 107, 337, 338, 660
tape drives
  host port speeds and, 134
  reports on, 305
tape ejection in DMF Manager, 232
tape management
  error reports, 443
  merging sparse tapes, 443
  merging sparse volumes, 543
Tape Management Facility
  See "TMF", 13
tape merging, 307
  See "volume merging", 543
tape mounting services
  See “OpenVault” or “TMF”, 13
tape MSP end of life, 784

tape MSP/LS and dmatread, 567
tape performance, 781
tape recall, 185
tapesize keyword, 559
tar file recall, 643
task, 53
TASK_GROUPS, 176, 293, 301, 358, 395, 409, 422, 453, 458, 464, 470
taskgroup, 113
taskgroup object
  overview, 272
  parameters, 304, 442
Tasks panel in DMF Manager, 193
tcpmux, 108
tcpmux service_limit error, 670
terminology, 14
theory of archiving, 19
third-party backup package configuration, 767
THRESHOLD, 307, 331, 443
threshold keyword, 559
tiered-storage management, 5
Time Navigator, 769
time synchronization, 124
time_expression configuration, 328
TIMEOUT_FLUSH, 107, 422
tmcollect, 675
tmconfig, 594
TMF
  availability, 170
  considerations, 170
  license, 74
  LS drive groups and, 505
  support for, 13
  tracing, 126
TMF drive disabling, 594
TMF_TMMNT_OPTIONS, 350, 409
TMP_DIR, 45, 102, 163, 285, 486
tmstat, 594
tools, 56
totals, 242
tpcrdm.dat, 647
tpcrdm.dat file, 537, 647
tpcrdm.key1.keys, 647
tpcrdm.key1.keys file, 537, 647
tpcrdm.key2.keys, 647
tpcrdm.key2.keys file, 537, 647
tpvrdm, 537
tpvrdm.dat, 538, 647
tpvrdm.dat file, 647
tpvrdm.vsn.keys, 538, 647
tpvrdm.vsn.keys file, 647
trace_directory, 126
trace_file_size, 126
trace_save_directory, 126
tracing and TMF, 126
trailing scaling character, 275, 385
transaction processing, 38
transports, 42
trickle mode, 314
troubleshooting, 641
  reporting problems to SGI, 675
ts tape driver, 337, 338
  drives not claimed, 660
  HBA drivers and initrd, 107
  tsreport, 66
tsmt, 671
tsreport, 305, 338
TSREPORT_OPTIONS, 331
tsset, 671
TYPE, 279, 290, 295, 300, 312, 343, 348, 352, 366, 367, 391, 393, 397, 411, 428, 433–435, 449, 456, 461, 468

U

uid expression, 382
ULTRIUM, 276
Ultrium5-1500, 631
Ultrium6-2500, 631
undocumented commands, 130
unit measures and DMF Manager, 234, 236

UNIX special files, 27
unknown mount option, 660
unmigrating file, 15
unsupported commands, 130
upage keyword, 559
update directive, 518, 545, 555
update keyword, 559
Update License button, 791
UPDATE_CTIME, 148, 358
updateage, 521
updatetime, 521, 524
upgrade best practices, 84
URL for DMF Manager, 10
USE_AS_DUMP_DEST, 359
USE_UNIFIED_BUFFER, 359
user interface commands, 57
user library (libdmfusr.so)
  archive file requests, 712
  context manipulation subroutines, 705
  copy file requests, 710
  distributed commands, 681
  DmuAllErrors_t, 686
  DmuAttr_t, 687
  DmuAwaitReplies(), 728
  DmuByteRange_t, 688
  DmuByteRanges_t, 689
  DmuChangedDirectory(), 707
  DmuCompletion_t, 692
  DmuCopyAsync_2(), 710
  DmuCopyRange_t, 693
  DmuCopyRanges_t, 693
  DmuCopySync_2(), 710
  DmuCreateContext(), 705
  DmuDestroyContext(), 707
  DmuErrHandler_f, 694
  DmuErrInfo_t, 695
  DmuError_t, 695
  DmuEvents_t, 696
  DmuFhandle_t, 696
  DmuFilesysInfo(), 708
  DmuFsysInfo_t, 696
  DmuFullRegbuf_t, 697
  DmuFullRegion_t, 698
  DmuFullstat_2_t, 699
  DmuFullstat_t, 698
  DmuFullstatByFhandleAsync(), 714
  DmuFullstatByFhandleSync(), 714
  DmuFullstatByPathAsync(), 714
  DmuFullstatByPathSync(), 714
  DmuFullstatCompletion(), 729
  DmuGetByFhandleAsync_2(), 720
  DmuGetByFhandleSync_2(), 720
  DmuGetByPathAsync_2(), 720
  DmuGetByPathSync_2(), 720
  DmuGetNextReply(), 730
  DmuGetThisReply(), 732
  DmuOperandType_t, 700
  DmuPriority_t, 701
  DmuPutByFhandleAsync(), 717
  DmuPutByFhandleSync(), 717
  DmuPutByPathAsync(), 717
  DmuPutByPathSync(), 717
  DmuRegion_t, 701
  DmuRegionbuf_t, 701
  DmuReplyOrder_t, 702
  DmuReplyType_t, 702
  DmuSetprojidByPathAsync, 725
  DmuSettagByFhandleAsync(), 725
  DmuSettagByFhandleSync(), 725
  DmuSettagByPathAsync(), 725
  DmuSettagByPathSync(), 725
  DmuSeverity_t, 702
  DmuVolGroup_t, 703
  DmuVolGroups_t, 703
  file request subroutines, 709
  fullstat requests, 714
  get file requests, 720
  IRIX considerations, 684
  library versioning, 684
  put file requests, 717
  request completion subroutines, 727
  Setprojid file requests, 723

Settag file requests, 725
sitelib.so and, 739
update in DMF 3.1, 784
user-accessible API subroutines for libdmfusr.so.2, 705
/usr/dmf/dmbase, 783
/usr/lib/dmf/dmf_client_ports, 188
/usr/lib/dmf/support/dmanytag, 785
/usr/lib/dmf/support/dmcleardcmtag, 785
/usr/lib/dmf/support/dmclearpartial, 786
/usr/lib/dmf/support/dmcleartag, 785
/usr/lib/dmf/support/manypartial, 786
/usr/sbin/lk_hostid, 74
/usr/share/doc/dmf-*/info/sample, 736

V

VALID_ROOT_HOSTS, 285
/var/lib/pcp-storage/archives, 130
/var/lib/pcp-storage/archives directory, 130
/var/log/xinetd.log, 670
verification
automated task, 337
daemon database integrity, 337
dmmaint and, 789
License Info, 791
license keys, 75
run_audit.sh, 337
verify
disk MSPs, 579
verify directive, 545, 555
VERIFY_POSITION, 409
version keyword, 559
VG, 244
configuration, 388, 389
objects, 273
selection for migrating files, 374, 380
terminology, 41
VG and COPAN, 110
vgnames, 551
vista.taf file, 530
VOL record
messages, 679
VOL records, 37, 533, 537
backup, 646
VOL_MSG_TIME, 422, 435
volgrp, 549
volgrp keyword, 559
volume group, 244
volume merge stopping, 307
volume merging
configuration of automated task, 443
LS, 543
stopping automatically, 443
terminology, 38
volume-to-volume merging, 543
VOLUME_GROUPS, 102, 409
VOLUME_LIMIT, 307, 332, 443
volumegroup, 116, 140
volumegroup object
associated task scripts, 308
overview, 273
parameters, 411
volumes, 244
Volumes panel in DMF Manager, 193, 229
vsn, 549
VSN on COPAN MAID, 64
vsnlist expression, 556
VTL
configuration best practices, 109
See "SGI 400 VTL", 109

W

WATCHER, 395
web service definition language, 600
web-based tool, 10
WEIGHT, 433
weighting of files for migration, 371, 378, 386
wfage keyword, 560

wfdate keyword, 560
What Is help, 201
when clause, 380
Windows client, 12
Windows Explorer, 12
Windows Explorer delay icon, 127
Windows Explorer hangs, 669
WRITE_CHECKSUM, 409, 453, 458, 464, 470
writeage, 549
writedate, 549
WSDL, 600

X

XDSM standard, 39
XFS, 8
XFS immutable bit, 140
xfsdump, 643
xfsrestore, 643
xinetd, 108, 165, 166
XVM failover, 49

Y

YAST and configuring network services, 83

Z

zone size and performance, 105
ZONE_SIZE, 105, 107, 423, 781
zoneblockid, 549
zonenumber, 549
zonepos, 549
zones, 534
zonesize, 680
zoning of the SAN switch, 49