HPSS Admin Guide

July 15, 2026: High Performance Storage System 12.1

Table of Contents

1. Release 12 . . . 3
   1.1. New features in 12.1 . . . 3
      1.1.1. SSM Command Line Interface (SSM CLI) . . . 3
      1.1.2. Limited Life-Time Session Tokens for S3 Proxy . . . 3
      1.1.3. Support for LTO-10 Enterprise Media . . . 3
      1.1.4. Encrypted Data Transfers . . . 4
      1.1.5. License Key for HPSS Core Servers . . . 4
      1.1.6. Mover control channel encryption . . . 4
      1.1.7. Port Range for Client API Applications . . . 4
   1.2. Features retired in 12.1 . . . 4
   1.3. Features deprecated in 12.1 . . . 5
   1.4. Changes to existing HPSS features in 12.1 . . . 5
      1.4.1. Updates to the SSM API . . . 5
   1.5. API changes in HPSS 12.1 . . . 5
      1.5.1. Changes to PIO . . . 5
      1.5.2. HPSS Client API Python Binding Changes (hpsspy) . . . 5
2. Release 11 . . . 6
   2.1. New features in 11.3 . . . 6
      2.1.1. Mark PVR Disabled . . . 6
      2.1.2. Persistent Asynchronous Stage Requests . . . 6
      2.1.3. UNIX Keytab Shadow File . . . 6
   2.2. HPSS Config History . . . 7
      2.2.1. New S3 Proxy Headers and Configuration . . . 8
      2.2.2. SSM API . . . 8
      2.2.3. Web SSM UI . . . 9
      2.2.4. Support for LTO-10 . . . 9
   2.3. Features retired in 11.3 . . . 9
   2.4. Features deprecated in 11.3 . . . 9
   2.5. Changes to existing HPSS features in 11.3 . . . 10
      2.5.1. HPSS GateKeeper Attributes . . . 10
      2.5.2. Apply Config Logging . . . 10
   2.6. API changes in HPSS 11.3 . . . 10
      2.6.1. Stage Callback Changes . . . 10
      2.6.2. User Attribute Changes . . . 11
      2.6.3. New Python API Bindings . . . 11
      2.6.4. Tape flag for library down . . . 11
      2.6.5. HPSS DB diagnostics tool . . . 12
   2.7. New features in 11.2 . . . 12

2.7.1. Client Striped IOD . . . 12
2.7.2. LibAIO Support . . . 12
2.7.3. S3 Kerberos Support . . . 12
2.8. Features retired in 11.2 . . . 13
2.9. Features deprecated in 11.2 . . . 13
2.10. Changes to existing HPSS features in 11.2 . . . 13
2.11. API changes in HPSS 11.2 . . . 13
2.11.1. Simplified PIO Interface . . . 13
2.11.2. Client Striped IOD in PIO . . . 14
2.12. New features in 11.1 . . . 14
2.12.1. HPSS Recurse tool . . . 14
2.12.2. VV Selection Priority (CR 411) . . . 14
2.12.3. Purge By Largest File First . . . 15
2.12.4. Secondary Data Copies in AWS . . . 15
2.13. Features retired in 11.1 . . . 15
2.14. Features deprecated in 11.1 . . . 15
2.15. Changes to existing HPSS features in 11.1 . . . 16
2.15.1. Updates to RTM Summary List window and rtmu tool . . . 16
2.16. API changes in HPSS 11.1 . . . 16
3. Release 10 . . . 17
3.1. New features in 10.3 . . . 17
3.1.1. TS1170 / 3592-70F/S / Jag7 support . . . 17
3.1.2. dumpv_pvl can display the PVR name . . . 17
3.1.3. Migration and Purge State Changes . . . 17
3.1.4. HPSS S3 interface . . . 17
3.1.5. Read Queue APIs . . . 17
3.1.6. Persisted Read Queues . . . 18
3.1.7. Updates to avoid server restarts . . . 18
3.1.8. Added purge filters based on file size and age (CR 558) . . . 18
3.2. New features in 10.2 . . . 18
3.2.1. Purge on migrate (CR 231) . . . 18
3.2.2. Reinit SCSI PVR control paths (CR 609) . . . 19
3.3. New features in 10.1 . . . 19
3.3.1. HPSS visualization and monitoring (CR 410) . . . 19
3.3.2. Restricted access (CR 625) . . . 19
3.3.3. Files now show the type of media in extended attributes . . . 19
3.3.4. Files now show current activity in extended attributes . . . 20
3.3.5. Low overhead read interface (lori - CR 562) . . . 20
3.3.6. dump_acct_sum will now dump the bandwidth table . . . 20
3.3.7. dumpv_pvl now displays the HPSS label format . . . 20
3.3.8. lscos and lsvol now support JSON output . . . 20

3.3.9. Toggle RAO on/off per tape device . . . 20
3.3.10. Repack will log media access INFO logs . . . 20
3.3.11. HPSS server metrics tool . . . 20
3.3.12. HPSS DB metrics tool . . . 21
3.4. Features retired in 10.1 . . . 21
3.4.1. Restricted user list . . . 21
3.5. Features deprecated in 10.1 . . . 21
3.6. Changes to existing HPSS features in 10.1 . . . 21
3.6.1. HPSS now supports 65535 alternate groups . . . 21
3.6.2. hpssmsg now supports more fields . . . 21
3.6.3. Migration/Purge server raw report files now using JSON Lines. . . . 21
3.6.4. JSON timestamps now in ISO 8601 format . . . 22
3.6.5. Parallel quaid file scans . . . 22
3.6.6. Avoid purging newly staged data with Last Access Time purge policies . . . 22
3.7. API changes in HPSS 10.1 . . . 22
4. Release 9 . . . 24
4.1. New features in 9.3 . . . 24
4.1.1. HPSS abort I/O tool (CR 546) . . . 24
4.1.2. Calculate the atime . . . 24
4.1.3. Support for Mover for fast positioning . . . 24
4.1.4. Parallel transfer abort in PFTP . . . 25
4.1.5. Defer tape labeling on import (CR 607) . . . 25
4.1.6. Updates to hpss_log_archive . . . 25
4.1.7. Disk Mover & VV I/O Metrics . . . 26
4.1.8. New location for HPSS man pages . . . 26
4.2. Features retired in 9.3 . . . 26
4.3. Features deprecated in 9.3 . . . 26
4.4. Changes to existing HPSS features in 9.3 . . . 26
4.4.1. Increased COS and storage class limits . . . 27
4.4.2. JSON output for dump_sspvs . . . 27
4.4.3. Users can now pass a VV name to dump_sspvs . . . 27
4.4.4. Pre-queued stages for mass stage . . . 27
4.4.5. PFTP now supports direct I/O . . . 27
4.4.6. Detect and avoid drives being cleaned . . . 27
4.5. New features in 9.2 . . . 28
4.5.1. Support for Spectra Logic TAOS . . . 28
4.5.2. Support for LTO-9 . . . 28
4.5.3. Increase tape aggregation limits . . . 28
4.6. Features retired in 9.2 . . . 29
4.7. Features deprecated in 9.2 . . . 29
4.8. Changes to existing HPSS features in 9.2 . . . 29

4.9. New features in 9.1 . . . 29
4.9.1. Provide API tools . . . 30
4.9.2. Automatic Change Class of Service . . . 30
4.9.3. Client I/O abort . . . 31
4.9.4. Account management in PFTP . . . 31
4.9.5. Buffered Tape Marks . . . 32
4.10. Features retired in 9.1 . . . 32
4.11. Features deprecated in 9.1 . . . 32
4.12. Changes to existing HPSS features in 9.1 . . . 32
4.12.1. The scsi.diag can now be squelched system-wide . . . 32
5. HPSS basics . . . 34
5.1. Introduction . . . 34
5.2. HPSS capabilities . . . 34
5.2.1. Network-centered architecture . . . 34
5.2.2. High data transfer rate . . . 34
5.2.3. Parallel operation . . . 35
5.2.4. Based on standard components . . . 35
5.2.5. Data integrity through transaction management . . . 35
5.2.6. Multiple hierarchies and Classes of Services . . . 35
5.2.7. Storage subsystems . . . 36
5.3. HPSS components . . . 36
5.3.1. HPSS files, filesets, volumes, storage segments and related metadata . . . 38
5.3.2. HPSS servers . . . 40
5.3.3. HPSS storage subsystems . . . 44
5.3.4. HPSS infrastructure . . . 44
5.3.5. HPSS user interfaces . . . 46
5.3.6. HPSS management interfaces . . . 47
5.3.7. HPSS policy modules . . . 48
5.4. HPSS hardware platforms . . . 49
5.4.1. Server platforms . . . 50
5.4.2. Client platforms . . . 50
5.4.3. Mover platforms . . . 50
6. HPSS planning . . . 51
6.1. Overview . . . 51
6.1.1. HPSS system architecture . . . 51
6.1.2. HPSS configuration planning . . . 52
6.1.3. Purchasing hardware and software . . . 54
6.1.4. HPSS operational planning . . . 54
6.1.5. HPSS deployment planning . . . 55
6.2. Requirements and intended uses for HPSS . . . 55
6.2.1. Storage system capacity . . . 56

6.2.2. Required throughputs . . . 56
6.2.3. Load characterization . . . 56
6.2.4. Usage trends . . . 56
6.2.5. Duplicate file policy . . . 57
6.2.6. Charging policy . . . 57
6.2.7. Security . . . 57
6.2.7.1. Cross realm access . . . 58
6.2.8. HPSS availability options . . . 58
6.3. Prerequisite software considerations . . . 59
6.3.1. Prerequisite software overview . . . 59
6.3.1.1. DB2 . . . 59
6.3.1.2. OpenSSL . . . 59
6.3.1.3. Kerberos . . . 59
6.3.1.4. LDAP and IBM Kerberos . . . 59
6.3.1.5. Java . . . 60
6.3.1.6. Use of libTI-RPC . . . 60
6.3.1.7. Jansson . . . 60
6.3.1.8. STK Tools . . . 60
6.3.1.9. Apache Commons Codecs . . . 60
6.3.1.10. nginx . . . 60
6.3.1.11. OIDC Provider . . . 60
6.3.2. Prerequisite summary By HPSS node type . . . 61
6.3.2.1. HPSS server nodes . . . 61
6.3.2.2. HPSS client nodes . . . 61
6.4. Hardware considerations . . . 62
6.4.1. Network considerations . . . 62
6.4.2. Robotically mounted tape . . . 63
6.4.2.1. Drive-controlled LTO libraries (IBM, Spectralogic) . . . 63
6.4.2.2. Oracle StorageTek . . . 63
6.4.2.3. Oracle StorageTek tape libraries that support ACSLS . . . 63
6.4.3. Manually mounted tape . . . 64
6.4.4. Tape devices . . . 64
6.4.4.1. Multiple media support . . . 64
6.4.5. Disk devices . . . 67
6.4.6. AWS Tape Gateway . . . 68
6.4.7. Special bid considerations . . . 68
6.5. HPSS sizing considerations . . . 68
6.5.1. HPSS user storage space . . . 69
6.5.2. HPSS infrastructure storage space . . . 69
6.5.2.1. HPSS and DB2 file systems . . . 72
6.5.2.2. HPSS metadata space . . . 75

6.5.2.3. HPSS file systems . . . 77
6.5.3. System memory and disk space . . . 79
6.5.3.1. Operating system disk spaces . . . 79
6.5.3.2. System disk space requirements for running SSM . . . 79
6.5.3.3. System memory and paging space requirements . . . 80
6.6. HPSS interface considerations . . . 80
6.6.1. Client API . . . 80
6.6.2. FTP . . . 81
6.6.3. Parallel FTP . . . 81
6.7. HPSS server considerations . . . 82
6.7.1. Core Server . . . 82
6.7.2. Migration/Purge Server . . . 85
6.7.3. Gatekeeper . . . 87
6.7.4. PVL . . . 89
6.7.5. PVR . . . 89
6.7.5.1. STK PVR . . . 89
6.7.5.2. Operator PVR . . . 90
6.7.5.3. SCSI PVR . . . 90
6.7.6. Mover . . . 90
6.7.6.1. Tape devices . . . 90
6.7.6.2. Disk devices . . . 91
6.7.6.3. Performance . . . 91
6.7.7. Logging service . . . 92
6.7.8. Startup Daemon . . . 92
6.7.9. Storage System Management . . . 93
6.8. Storage subsystem considerations . . . 95
6.9. Storage policy considerations . . . 95
6.9.1. Migration policy . . . 95
6.9.1.1. Migration policy for disk . . . 96
6.9.1.2. Migration policy for tape . . . 96
6.9.2. Purge policy . . . 96
6.9.3. Accounting policy and validation . . . 97
6.9.4. Security policy . . . 99
6.9.4.1. Client API . . . 99
6.9.4.2. FTP/PFTP . . . 100
6.9.4.3. Name space . . . 100
6.9.4.4. Security audit . . . 100
6.9.5. Logging policy . . . 100
6.9.6. Gatekeeping . . . 101
6.10. Storage characteristics considerations . . . 103
6.10.1. Storage class . . . 104

6.10.1.1. Media block size selection . . . 104
6.10.1.2. Virtual volume block size selection (disk) . . . 104
6.10.1.3. Virtual volume block size selection (tape) . . . 104
6.10.1.4. Stripe width selection . . . 105
6.10.1.5. Blocks between tape marks selection (tape only) . . . 107
6.10.1.6. Minimum storage segment size selection (disk only) . . . 107
6.10.1.7. Maximum storage segment size selection . . . 108
6.10.1.8. Maximum VVs to write (tape only) . . . 108
6.10.1.9. Average number of storage segments (disk only) . . . 109
6.10.1.10. PV estimated size and PV size selection . . . 109
6.10.1.11. Optimum access size selection . . . 109
6.10.1.12. Some recommended parameter values for supported storage media . . . 109
6.10.2. Storage hierarchy . . . 113
6.10.3. Class of Service . . . 113
6.10.3.1. Selecting minimum file size . . . 114
6.10.3.2. Selecting maximum file size . . . 114
6.10.3.3. Selecting stage code . . . 114
6.10.3.4. Selecting optimum access size . . . 115
6.10.3.5. Selecting average latency . . . 115
6.10.3.6. Selecting transfer rate . . . 116
6.10.3.7. StripeLength and StripeWidth hints . . . 116
6.10.4. File families . . . 116
6.11. HPSS performance considerations . . . 116
6.11.1. DB2 . . . 117
6.11.2. Bypassing potential bottlenecks . . . 117
6.11.3. Configuration . . . 118
6.11.4. FTP/PFTP . . . 118
6.11.5. Client API . . . 119
6.11.6. Core Server . . . 119
6.11.7. Logging . . . 120
6.11.8. Cross-realm trust . . . 120
6.11.9. Gatekeeping . . . 120
6.11.10. HPSSFS-FUSE interface . . . 121
6.11.11. SSM API . . . 121
6.11.12. Web SSM UI . . . 121
6.12. HPSS metadata backup considerations . . . 121
6.13. HPSS security considerations . . . 122
6.14. HPSS Disk Mover and VV Load-Balancing . . . 122
6.14.1. Overview . . . 122
6.14.2. Disk Mover & VV I/O Metrics . . . 122
6.14.3. Maintaining Disk I/O Operations Metrics Files . . . 124

6.14.4. JSON format . . . 125
6.14.4.1. Disk Allocation . . . 125
6.14.4.2. Disk I/O Begin & Completion . . . 126
6.14.5. Monitoring Disk I/O Operations Metrics . . . 127
7. System preparation . . . 129
7.1. General setup . . . 129
7.2. Set up file systems . . . 130
7.2.1. DB2 file system . . . 130
7.2.2. HPSS file system . . . 131
7.3. Set up tape libraries . . . 131
7.3.1. Oracle StorageTek . . . 131
7.3.2. SCSI . . . 132
7.4. Verify tape drives . . . 133
7.4.1. Linux . . . 133
7.5. Set up disk drives . . . 134
7.5.1. Linux . . . 134
7.6. Set up network parameters . . . 134
7.6.1. HPSS.conf configuration file . . . 136
7.7. Port mapping and firewall considerations . . . 137
7.7.1. Web SSM and SSM API . . . 138
7.8. Semaphore values . . . 138
7.9. Enable Core Dumps . . . 140
8. HPSS installation and infrastructure configuration . . . 142
8.1. Prepare for installation . . . 142
8.1.1. Distribution media . . . 142
8.1.2. Software installation packages . . . 142
8.1.3. Create owner account for HPSS files . . . 143
8.1.4. Installation target directory preparation . . . 143
8.2. Install prerequisite software . . . 143
8.2.1. Install Java . . . 143
8.2.2. Install Jansson . . . 144
8.2.3. Install TI-RPC . . . 144
8.2.4. Install Ncurses . . . 144
8.2.5. Install MIT Kerberos . . . 144
8.2.6. Install LDAP (if using LDAP authorization) . . . 144
8.2.7. Install DB2 and set up permanent license . . . 145
8.2.8. Install HPSS License . . . 145
8.2.9. Install OIDC Prerequisites . . . 146
8.2.10. Install Web Service Prerequisites . . . 146
8.3. Install HPSS with RPMs . . . 146
8.4. Install HPSS . . . 148

8.4.1. On core . . . 148
8.4.2. On Mover . . . 148
8.4.3. On client . . . 149
8.4.4. On remote PVR . . . 149
8.4.5. Generate and bind the DB2 helper program . . . 149
8.4.6. Install SSL certificates for web services . . . 150
8.4.7. Update default DB2 link . . . 150
8.5. Configure OIDC Provider . . . 151
8.5.1. Configuring Keycloak . . . 151
8.5.2. Configuring Your OIDC Provider for HPSS . . . 152
8.6. Configure HPSS infrastructure . . . 153
8.6.1. Navigating and general **mkhpss** behavior . . . 153
8.6.2. Configure HPSS - root subsystem machine . . . 154
8.6.2.1. Pre-installation configuration . . . 154
8.6.2.2. Configure HPSS security services . . . 156
8.6.2.3. Configure DB2 services . . . 168
8.6.2.4. *Setting up off-node DB2* . . . 174
8.6.2.5. Configure other services . . . 180
8.6.2.6. Configure Logging . . . 182
8.6.2.7. Create configuration bundle . . . 182
8.6.3. Configure HPSS - secondary subsystem machine . . . 183
8.6.4. Troubleshooting **mkhpss** . . . 184
8.7. Post-installation procedures . . . 185
8.8. Locate HPSS documentation and set up manual pages . . . 186
8.8.1. Documentation and SSM help package . . . 186
8.8.2. Manual pages setup . . . 186
8.9. Define HPSS environment variables . . . 187
8.10. Set up a remote PVR . . . 188
8.11. Tune DB2 . . . 190
8.12. Supporting both UNIX and Kerberos authentication for SSM . . . 190
8.13. HPSS IPv6 support . . . 193
8.13.1. Usage examples . . . 193
9. Installation and configuration of the Elastic (ELK) Stack . . . 196
9.1. Installing the ELK stack . . . 196
9.1.1. Install Filebeat . . . 196
9.1.2. Install Logstash . . . 197
9.1.3. Install Elastic . . . 197
9.1.4. Install Kibana . . . 198
9.1.5. Scaling Elastic . . . 198
9.2. Installing the HPSS data capture components . . . 198
9.2.1. HPSS data capture scripts . . . 198

9.2.2. How to configure the data capture scripts . . . 199
9.3. Setup the HPSS dashboards . . . 200
9.3.1. Load the HPSS templates into Kibana . . . 200
9.3.2. The HPSS dashboards . . . 200
9.4. Captured data . . . 201
10. HPSS Rumbler . . . 203
10.1. Introduction . . . 203
10.2. HPSS Rumbler Package Install . . . 203
10.3. Database Preparation . . . 203
10.3.1. Shutdown HPSS and Backup Databases . . . 203
10.3.2. Database Backups . . . 204
10.4. Db2 History Table Configuration . . . 204
10.4.1. Add History Tables . . . 204
10.4.2. Update Db2 Registry . . . 204
10.4.3. Update Db2 Timestamp Settings . . . 205
10.4.4. Update Db2 Authorization for Access to Stored Procedures and Functions . . . 205
10.4.5. Db2 Runstats and Maintenance . . . 205
10.4.6. Db2 Backups . . . 206
10.4.7. Verifying History Setup . . . 206
10.5. Rumbler Usage . . . 207
10.5.1. Full Run . . . 207
10.5.2. Partial Run . . . 208
10.6. History Table Maintenance . . . 210
10.6.1. Overview . . . 210
10.6.2. Additional Setup for Manage Partitioning Tool . . . 211
10.6.2.1. ibm_db and pycrypto modules . . . 211
10.6.2.2. HPSS Client *hpss* module . . . 211
10.6.2.3. Edit the udf.conf file . . . 211
10.6.3. Configuration Files . . . 212
10.6.4. History Management Tools . . . 213
10.7. Viewing History Table Partitions . . . 213
10.7.1. Table Pruning . . . 215
10.7.2. Adding Partitions to Tables . . . 215
11. HPSS S3 interface . . . 217
11.1. Overview . . . 217
11.1.1. End User Authentication . . . 218
11.1.1.1. Password Database . . . 218
11.2. HPSS S3 Interface Setup . . . 219
11.3. Interoperation . . . 223
11.4. HPSS Specific Options . . . 224
11.5. S3 Multipart Upload and HPSS File Handles . . . 225

11.6. S3 Object Listing . . . 225
11.7. S3 Virtual Hosting . . . 225
11.8. S3 Buckets . . . 225
11.9. S3 Client Setup . . . 226
11.9.1. General . . . 226
11.9.2. s3cmd . . . 226
11.9.3. boto3 . . . 227
11.10. Unsupported Operations . . . 227
11.11. Limitations . . . 228
11.12. Performance . . . 229
11.13. Scaling and Load Balancing . . . 229
11.14. Enabling HTTPS for S3 proxy . . . 230
11.15. Setting up for limited lifetime session tokens . . . 231
11.15.1. Configuring the OIDC provider . . . 231
11.15.2. Configuring the S3 API server . . . 232
11.15.3. Enabling support for the session token GSS mechanism . . . 234
11.15.4. Configuring the HPSS PFTP daemon . . . 234
11.15.5. Obtaining a session token . . . 235
11.15.6. Login helper . . . 237
11.15.6.1. Headless Mode . . . 237
11.15.7. Managing sessions . . . 238
11.15.7.1. Session management configuration . . . 239
11.15.8. Integrating Keycloak with existing systems . . . 240
11.16. Setting up for Kerberos use . . . 240
12. HPSS configuration overview . . . 242
12.1. Introduction . . . 242
12.2. Starting the SSM GUI for the first time . . . 242
12.3. HPSS configuration roadmap (new HPSS sites) . . . 243
12.4. Initial HPSS startup roadmap (all sites) . . . 244
12.5. Additional configuration roadmap (all sites) . . . 245
12.6. Verification checklists (all sites) . . . 245
12.6.1. Configuration checklists . . . 245
12.6.2. Operational checklists . . . 246
12.6.3. Performance checklist . . . 247
13. Security and system access . . . 248
13.1. Security services . . . 248
13.1.1. Security services configuration . . . 248
13.1.2. Security mechanisms . . . 249
13.1.2.1. UNIX . . . 249
13.1.2.2. UNIX and NIS or LDAP . . . 250
13.1.2.3. Kerberos 5 . . . 250

13.1.2.4. LDAP . . . 250
13.2. HPSS server security ACLs . . . 253
13.3. SSM user security . . . 254
13.4. Restricting user access to HPSS . . . 255
13.4.1. Overview . . . 255
13.4.2. Configuration File . . . 255
13.4.2.1. File Permissions . . . 255
13.4.2.2. File Format . . . 255
13.4.3. ra_verify_config . . . 256
13.4.4. Refresh Restricted Access Table . . . 256
13.4.5. Restriction Types . . . 256
13.4.6. Example Restrict Access File . . . 256
14. Using SSM . . . 258
14.1. The SSM System Manager . . . 258
14.1.1. Starting the SSM System Manager . . . 258
14.1.2. Tuning the System Manager RPC thread pool and request queue sizes . . . 258
14.1.3. Labeling the System Manager RPC program number . . . 259
14.2. Configuration and startup of the SSM API and Web SSM UI . . . 260
14.2.1. Overview . . . 260
14.2.2. Configuring nginx for Reverse Proxying . . . 262
14.2.3. Quick startup of the SSM API . . . 262
14.2.4. SSM API Configuration Details . . . 264
14.2.5. SSM API Docs Endpoint . . . 264
14.2.5.1. SSM API Docs and Versioning . . . 265
14.2.6. Client interface setup . . . 265
14.2.7. Creating SSM API Users . . . 265
14.2.8. Creating API Access Tokens . . . 266
14.2.9. Examples: Using the SSM API . . . 266
14.2.10. Managing SSM API Sessions and API Keys . . . 267
14.2.11. Current limitations . . . 268
14.2.12. Quick startup of the Web SSM UI . . . 268
14.2.13. Using the Web SSM UI . . . 268
14.3. Quick startup of the SSM CLI . . . 270
14.3.1. Install or Update the SSM CLI . . . 270
14.3.2. Configuring the SSM CLI . . . 271
14.3.3. Configuration settings for the SSM CLI . . . 272
14.3.4. Environment Variables for the SSM CLI . . . 273
14.3.5. Command-line Option for the SSM CLI . . . 273
14.3.6. HTTP configuration for the SSM CLI . . . 274
14.3.7. Auto-Completion in the SSM CLI . . . 275
14.4. Quick startup of **hpssgui** . . . 276

14.5. Configuration and startup of **hpssgui** and **hpssadm** . . . 277
14.5.1. Configuring the System Manager authentication for SSM clients . . . 278
14.5.2. Creating the SSM user accounts . . . 278
14.5.2.1. The **hpssuser** utility . . . 278
14.5.2.2. SSM user authorization . . . 279
14.5.2.3. User keytabs . . . 280
14.5.3. SSM configuration file . . . 281
14.5.3.1. login.conf . . . 283
14.5.3.2. krb5.conf (for use with Kerberos authentication only) . . . 284
14.5.4. SSM help files (optional) . . . 284
14.5.5. SSM desktop client packaging . . . 285
14.5.5.1. Automatic SSM client packaging and installation . . . 285
14.5.5.2. Manual SSM client packaging and installation . . . 286
14.5.6. Using SSM through a firewall . . . 287
14.5.6.1. The firewall problem . . . 287
14.5.6.2. Solutions for operating through a firewall . . . 287
14.5.6.3. Example: Using **hpssgui** through a firewall . . . 288
14.6. Multiple SSM sessions . . . 290
14.7. SSM window conventions . . . 290
14.8. Common window elements . . . 292
14.9. Help menu overview . . . 295
14.10. Monitor, Operations, and Configure menus overview . . . 296
14.10.1. Monitor menu . . . 296
14.10.2. Operations menu . . . 298
14.10.3. Configure menu . . . 300
14.11. SSM-specific windows . . . 301
14.11.1. HPSS Login . . . 302
14.11.2. About HPSS . . . 303
14.11.3. HPSS Health and Status . . . 304
14.11.3.1. SM server connection status indicator . . . 306
14.11.3.2. HPSS status . . . 306
14.11.3.3. HPSS statistics . . . 308
14.11.3.4. Menu tree . . . 309
14.11.3.5. File menu . . . 309
14.11.3.6. View menu . . . 310
14.11.4. SSM information windows . . . 310
14.11.4.1. System Manager Statistics window . . . 310
14.11.4.2. User Session Information window . . . 315
14.12. SSM list preferences . . . 316
15. Global and subsystem configuration . . . 320
15.1. Global Configuration window . . . 320

15.2. Storage subsystems . . . 325
15.2.1. Subsystems list window . . . 325
15.2.2. Creating a new storage subsystem . . . 326
15.2.3. Storage Subsystem Configuration window . . . 327
15.2.3.1. Create storage subsystem metadata . . . 330
15.2.3.2. Create storage subsystem configuration . . . 330
15.2.3.3. Create storage subsystem servers . . . 331
15.2.3.4. Assign a Gatekeeper if required . . . 331
15.2.3.5. Assign storage resources to the storage subsystem . . . 331
15.2.3.6. Create storage subsystem fileset and junction . . . 331
15.2.3.7. Migration and purge policy overrides . . . 331
15.2.3.8. Storage class threshold overrides . . . 332
15.2.4. Modifying a storage subsystem . . . 332
15.2.5. Deleting a storage subsystem . . . 332
15.3. HPSS trashcans . . . 333
15.3.1. The trashcan feature is on . . . 334
15.3.2. Trashcan operations for the administrator . . . 336
16. HPSS servers . . . 338
16.1. Server list . . . 338
16.2. Server configuration . . . 344
16.2.1. Common server configuration . . . 346
16.2.1.1. Basic controls . . . 346
16.2.1.2. Execution controls . . . 347
16.2.1.3. Interface controls . . . 349
16.2.1.4. Security controls . . . 352
16.2.1.5. Audit policy . . . 354
16.2.2. Core Server-specific configuration . . . 356
16.2.2.1. Additional Core Server configuration . . . 360
16.2.2.2. Core Server I/O limit configuration . . . 362
16.2.2.3. Core Server max open files configuration . . . 365
16.2.3. Gatekeeper-specific configuration . . . 367
16.2.4. Core Server additional configuration . . . 368
16.2.5. Core Server Behavior Under an Invalid License . . . 368
16.2.6. Migration/Purge Server (MPS)-specific configuration . . . 369
16.2.7. Mover-specific configuration . . . 370
16.2.7.1. Mover-specific configuration window . . . 370
16.2.7.2. Additional Mover configuration . . . 373
16.2.7.3. The Mover encryption cipher . . . 378
16.2.8. Physical Volume Repository (PVR)-specific configuration . . . 380
16.2.8.1. Operator PVR-specific configuration window . . . 380
16.2.8.2. SCSI PVR-specific configuration window . . . 382

16.2.8.3. STK PVR-specific configuration window . . . 386
16.2.9. RAIT Engine-specific configuration . . . 392
16.2.9.1. RAIT Engine-specific configuration window . . . 392
16.2.10. Deleting a server configuration . . . 395
16.3. Monitoring server information . . . 397
16.3.1. Basic Server Information . . . 397
16.3.2. Specific server information . . . 400
16.3.2.1. Core Server Information window . . . 400
16.3.2.2. Gatekeeper Information window . . . 412
16.3.2.3. Migration/Purge Server Information window . . . 416
16.3.2.4. Mover Information window . . . 416
16.3.2.5. Physical Volume Library (PVL) information window . . . 418
16.3.2.6. Operator PVR information window . . . 419
16.3.2.7. SCSI PVR information window . . . 421
16.3.2.8. STK PVR information window . . . 425
16.3.2.9. RAIT Engine Information window . . . 428
16.4. Real-Time Monitoring (RTM) . . . 429
16.4.1. RTM Summary List . . . 430
16.4.2. RTM Detail . . . 431
16.5. I/O Abort . . . 440
16.5.1. Aborting an I/O . . . 440
16.5.2. Expectations in Aborting I/O . . . 441
16.6. Starting HPSS . . . 442
16.6.1. Starting HPSS prerequisite software . . . 442
16.6.2. Starting HPSS servers . . . 442
16.6.2.1. Starting the Startup Daemons . . . 442
16.6.2.2. Starting SSM . . . 443
16.6.2.3. Starting other HPSS servers . . . 443
16.6.2.4. Automatic server restart . . . 444
16.7. Stopping HPSS . . . 444
16.7.1. Shutting down an HPSS server . . . 444
16.7.2. Shutting down all HPSS servers . . . 445
16.7.3. Halting an HPSS server . . . 445
16.7.4. Shutting down the SSM server . . . 446
16.7.5. Shutting down the Startup Daemon . . . 446
16.7.6. Stopping the prerequisite software . . . 446
16.8. Server repair and reinitialization . . . 446
16.8.1. Repairing an HPSS server . . . 447
16.8.2. Reinitializing a server . . . 447
16.9. Forcing an SSM connection . . . 450
17. Storage configuration . . . 451

17.1. Storage classes . . . 451
17.1.1. Configured Storage Classes window . . . 451
17.1.2. S3 Polling Intervals . . . 453
17.1.3. Disk Storage Class Configuration . . . 453
17.1.4. Tape Storage Class Configuration . . . 458
17.1.5. Object Store Storage Class Configuration . . . 465
17.1.6. Storage class subsystem thresholds . . . 470
17.1.6.1. Disk Storage Subsystem-Specific Thresholds . . . 470
17.1.6.2. Tape Storage Subsystem-Specific Thresholds . . . 473
17.1.7. Changing a storage class definition . . . 475
17.1.8. Deleting a storage class definition . . . 476
17.2. Storage hierarchies . . . 476
17.2.1. Hierarchies window . . . 476
17.2.2. Storage Hierarchy Configuration window . . . 478
17.2.3. Changing a storage hierarchy definition . . . 480
17.2.4. Deleting a storage hierarchy definition . . . 480
17.3. Classes of Service . . . 480
17.3.1. Classes of Service window . . . 480
17.3.2. Class of Service Configuration window . . . 482
17.3.3. Changing a Class of Service definition . . . 489
17.3.4. Deleting a Class of Service definition . . . 490
17.3.5. Changing a file's Class of Service . . . 490
17.3.6. Canceling a Class of Service change request . . . 491
17.4. Migration policies . . . 491
17.4.1. Migration Policies window . . . 491
17.4.2. Migration policy configuration . . . 493
17.4.2.1. Disk Migration Policy configuration . . . 493
17.4.2.2. Tape Migration Policy configuration . . . 499
17.4.2.3. Changing a migration policy . . . 502
17.4.2.4. Deleting a migration policy . . . 502
17.5. Purge policies . . . 503
17.5.1. Purge Policies window . . . 503
17.5.2. Purge Policy configuration . . . 504
17.5.3. Changing a purge policy . . . 508
17.5.4. Deleting a purge policy . . . 509
17.5.5. Purge parallelism . . . 509
17.6. File families . . . 509
17.6.1. File Family Configuration . . . 510
17.6.2. Changing a file family . . . 511
17.6.3. Deleting a file family . . . 511
18. Device and drive management . . . 512

18.1. Configure a new device and drive . . . 512
18.1.1. Devices and Drives window . . . 523
18.1.2. Enable variable block sizes for tape devices . . . 528
18.1.3. Changing a drive's configuration . . . 528
18.1.4. Deleting a drive's configuration . . . 529
18.2. Monitoring devices and drives . . . 530
18.2.1. Mover Device Information window . . . 531
18.2.2. PVL Drive Information window . . . 538
18.3. Changing device and drive state . . . 543
18.3.1. Unlocking a drive . . . 543
18.3.2. Locking a drive . . . 543
18.3.3. Repairing the state of a device or drive . . . 544
18.3.4. Resetting drive statistics . . . 544
18.3.5. Tape Drive Quota list window . . . 544
19. Volume and storage management . . . 546
19.1. Adding storage space . . . 546
19.1.1. Importing volumes into HPSS . . . 546
19.1.1.1. Import Tape Volumes window . . . 548
19.1.1.2. Selecting import type for tape cartridges . . . 552
19.1.1.3. Import Disk Volumes window . . . 553
19.1.1.4. Selecting import type for disk volumes . . . 557
19.1.1.5. Import Object Store Volumes window . . . 558
19.1.2. Creating storage resources . . . 562
19.1.2.1. Create Tape Resources window . . . 562
19.1.2.2. Prioritize Tape Resource Selection . . . 567
19.1.2.3. Aspects of Selection Priority Weights . . . 568
19.1.2.4. Create Disk Resources window . . . 569
19.1.2.5. Create Object Store Resources window . . . 573
19.2. Taking a tape library down for maintenance . . . 578
19.3. Removing storage space . . . 579
19.3.1. Deleting storage resources . . . 579
19.3.1.1. Rules for deleting resources . . . 579
19.3.1.2. Delete Resources window . . . 579
19.3.2. Exporting volumes from HPSS . . . 583
19.3.2.1. Rules for exporting volumes . . . 583
19.3.2.2. Export Volumes window . . . 583
19.4. Monitoring storage space . . . 587
19.4.1. Active Storage Classes window . . . 587
19.4.2. MPS Disk Storage Class Information . . . 592
19.4.3. MPS Tape Storage Class Information . . . 599
19.4.4. MPS Object Store Storage Class Information . . . 603

19.5. Dealing with a space shortage . . . 605
19.5.1. Manually starting migration . . . 605
19.5.2. Manually starting purge . . . 606
19.5.3. Repacking and reclaiming volumes . . . 606
19.5.4. Verifying data integrity . . . 608
19.5.4.1. Repack Virtual Volumes window . . . 609
19.5.4.2. Reclaim Virtual Volumes window . . . 611
19.6. Volume management . . . 612
19.6.1. Lookup Cartridges and Volumes window . . . 612
19.6.2. PVL Volume Information window . . . 613
19.6.3. PVR Cartridge Information window . . . 616
19.6.4. Core Server Volume and Segment windows . . . 620
19.6.4.1. Core Server Disk Volume information window . . . 620
19.6.4.2. Core Server Tape Volume information window . . . 626
19.6.4.3. Core Server Object Store Volume information window . . . 634
19.6.5. Changing Core Server volume condition . . . 639
19.6.6. Moving PVR cartridges to another PVR . . . 641
19.6.6.1. Move Cartridges window . . . 642
19.7. Monitoring and managing volume mounts . . . 645
19.7.1. PVL Job Queue window . . . 645
19.7.2. PVL Request Information window . . . 649
19.7.3. Canceling queued PVL requests . . . 652
19.7.4. Tape Check-In Requests window . . . 652
19.7.5. Tape Mount Requests window . . . 653
19.7.6. Administrative tape dismounts . . . 655
19.8. Storage technology replacement . . . 655
20. Logging and status . . . 658
20.1. Logging overview . . . 658
20.2. Log policies . . . 659
20.2.1. Creating a log policy . . . 659
20.2.2. Logging Policies window . . . 659
20.2.2.1. Logging Policy configuration window . . . 660
20.2.3. Changing a log policy . . . 663
20.2.4. Deleting a log policy . . . 663
20.3. Managing syslog . . . 663
20.3.1. Configuring syslog . . . 663
20.3.2. Recommendations for syslog configuration . . . 664
20.3.3. Interactions between HPSS and syslog . . . 665
20.3.4. Configure syslog message size . . . 665
20.3.5. Configure multi-part messages . . . 666
20.3.6. Restart Syslog After Configuration Change . . . 666

20.3.7. Managing rate-limiting and duplication . . . 667
20.3.8. Considerations For SELinux . . . 667
20.3.9. Managing log archiving . . . 668
20.3.10. Recommendations for logrotate configuration . . . 669
20.3.11. Viewing HPSS logs . . . 670
20.4. Managing SSM alarms and events . . . 671
20.4.1. Alarms and Events window . . . 671
20.4.2. Alarm/Event Information . . . 671
20.4.3. Diagnosing HPSS problems with alarms and events . . . 673
20.4.4. Controlling SSM log message handling . . . 674
20.4.4.1. Controlling the System Manager log message cache . . . 674
20.4.4.2. Controlling log messages displayed by **hpssgui** and **hpssadm** . . . 675
20.4.5. Media Access Logging . . . 676
20.5. Log files of various types - a comprehensive list . . . 678
20.6. Disk Mover & VV I/O Metrics . . . 683
21. Filesets and junctions . . . 684
21.1. Filesets & Junctions List . . . 684
21.2. Creating an HPSS fileset . . . 686
21.2.1. Create Fileset window . . . 687
21.3. Managing existing filesets . . . 689
21.3.1. Core Server Fileset Information window . . . 689
21.4. Deleting filesets . . . 692
21.5. Creating a junction . . . 692
21.5.1. Create Junction window . . . 693
21.6. Deleting a junction . . . 694
22. Files, directories and objects by SOID . . . 695
22.1. Files and Directories window . . . 695
22.1.1. File/Directory Information window . . . 696
22.2. Objects by SOID window . . . 704
23. Tape aggregation . . . 706
23.1. Overview of tape aggregation . . . 706
23.2. Tape aggregation performance considerations . . . 706
23.3. Ordered Migration and Full Aggregate Recall . . . 707
23.3.1. Ordered migration example - read-optimized aggregates . . . 708
23.4. Configuring tape aggregation . . . 708
23.5. Debugging tape aggregation . . . 708
23.6. Caveats . . . 708
24. Tape Ordered Recall (TOR) . . . 710
24.1. Overview of TOR . . . 710
24.2. Multiple User Considerations . . . 710
24.3. TOR ordering behavior . . . 711

24.4. TOR performance considerations . . . 711
24.4.1. TOR General Configuration Parameters . . . 712
24.4.2. Mass Parallel Access Configuration Parameters . . . 713
24.4.3. Batch Access Configuration Parameters . . . 714
24.5. Configuring TOR . . . 714
24.6. Monitoring Tape Order Recall . . . 715
24.7. Caveats . . . 715
24.8. HPSS Core Server Request Queue . . . 715
24.8.1. HPSS Core Server Request Queue Monitoring . . . 717
24.8.2. HPSS Core Server Request Queue Management . . . 717
24.8.3. Using Quaid and Lori . . . 717
25. User accounts and accounting . . . 719
25.1. Managing HPSS users . . . 719
25.1.1. Adding HPSS users . . . 719
25.1.1.1. Add all user ID types . . . 719
25.1.1.2. Add a user ID . . . 720
25.1.1.3. Add an SSM user ID . . . 721
25.1.2. Deleting HPSS users . . . 721
25.1.3. Listing HPSS users . . . 721
25.1.4. Create an SSM client package . . . 722
25.2. Accounting . . . 722
25.2.1. Accounting Policy window . . . 723
25.2.2. Accounting reports and status . . . 725
25.2.2.1. Generating an accounting report . . . 725
25.2.2.2. Accounting Status window . . . 726
25.2.2.3. Interpreting the accounting report . . . 727
25.2.3. Accounting procedures . . . 730
25.2.3.1. Site-defined accounting configuration files and procedures . . . 731
25.2.3.2. Accounting intervals and charges . . . 733
26. Object store accounts and accounting . . . 734
26.1. Managing object store accounts . . . 734
26.1.1. Object Store Accounts window . . . 734
26.1.2. Object Store Account Configuration . . . 735
26.2. Managing object store account identities . . . 735
26.2.1. Object Store Account Identities window . . . 736
26.2.2. Object Store Account Identity Configuration . . . 737
26.3. Managing Object Store Account Buckets . . . 738
26.3.1. Object Store Account Buckets window . . . 738
26.3.2. Object Store Account Bucket Configuration . . . 739
26.4. Managing Object Store Account Mappings . . . 740
26.4.1. Object Store Account Mappings window . . . 740

26.4.2. Object Store Account Mapping Configuration . . . 741
27. User interfaces . . . 743
27.1. Client API configuration . . . 743
27.2. HPSS UNIX Tools Configuration . . . 746
27.3. FTP/PFTP daemon configuration . . . 746
28. Backup and recovery . . . 752
28.1. HPSS metadata backup and recovery . . . 752
28.1.1. HPSS administrator responsibilities for DB2 . . . 752
28.1.2. Overview of the DB2 backup process . . . 753
28.1.2.1. Configuring DB2 for online backup . . . 754
28.1.3. Overview of the DB2 recovery process . . . 755
28.2. HPSS system environmental backup . . . 755
28.2.1. HPSS file system backup . . . 755
28.2.2. License file backup . . . 756
28.2.3. Operating system backup . . . 756
28.2.4. Kerberos backup . . . 756
28.3. HPSS user data recovery: Recovering HPSS files from damaged HPSS volumes . . . 757
28.4. DB2 monitoring . . . 759
28.5. DB2 space shortage . . . 760
28.5.1. DMS table spaces . . . 760
28.5.2. SMS table spaces . . . 761
29. Management tools . . . 762
29.1. Utility overview . . . 762
29.1.1. Fileset and junction management . . . 762
29.1.2. Disk management . . . 762
29.1.3. Tape management . . . 762
29.1.4. System info . . . 763
29.1.5. System management . . . 764
29.1.6. User interfaces . . . 766
29.1.6.1. HPSS UNIX Tools . . . 766
29.1.7. User-defined Attributes . . . 767
29.1.8. Testing and debugging . . . 767
29.1.9. Unsupported tools . . . 768
29.2. Tape queue monitoring . . . 769
29.2.1. Example . . . 769
29.2.2. JSON field descriptions . . . 770
29.3. HPSS ACL permission exceptions . . . 774
30. HPSS features . . . 777
30.1. End to End Data Integrity . . . 777
30.1.1. How to Enable LBP . . . 777
30.1.2. How to Enable File Hash . . . 777

30.1.3. Check your volumes with lbp_verify . . . 777
30.2. HPSS Config History . . . 778
30.3. Encryption in HPSS . . . 778
30.3.1. Client Data Encryption . . . 778
30.3.1.1. Enabling Client Data Encryption . . . 779
30.3.1.2. Client-Specific Encryption Setup . . . 779
30.3.2. Mover Data Encryption . . . 779
30.3.2.1. Configuring Mover Encryption . . . 779
30.3.2.2. Settings Propagation . . . 780
30.3.3. Control Traffic Encryption . . . 780
30.3.3.1. RPC Protection Levels . . . 780
30.3.3.2. Client RPC Configuration . . . 780
30.3.3.3. Web-Based Management Interface Encryption . . . 781
30.3.3.4. Additional Security Configuration . . . 781
Appendix A: Glossary of terms and acronyms . . . 782
Appendix B: References . . . 796
Appendix C: Developer acknowledgments . . . 798
Appendix D: `HPSS.conf` configuration file . . . 799
D.1. PFTP Client Stanza . . . 799
D.2. PFTP Client Interfaces Stanza . . . 805
D.3. Multinode Table Stanza . . . 808
D.4. Network Options Stanza . . . 811
D.5. PFTP Daemon Stanza . . . 816
D.6. Transfer Agent Stanza . . . 832
D.7. Stanzas reserved for future use . . . 840
31. HPSS Environment Variables . . . 841
31.1. HPSS Environment Variables . . . 841
31.2. HPSS Misc Environment . . . 841
31.3. HPSS Server Group Names . . . 842
31.4. HPSS Server Principal Names . . . 842
31.5. HPSS Server Principal UIDs . . . 843
31.6. HPSS Server Executable Names . . . 844
31.7. SSM Utility locations . . . 845
31.8. Logging unix files . . . 845
31.9. Accouting Unix files . . . 846
31.10. MPS Unix files . . . 846
31.11. Gatekeeper environment variables . . . 846
31.12. Database environment variables . . . 846
31.13. System environment variables . . . 847
31.14. SSM Specific . . . 848
31.15. API Specific . . . 851

31.16. Core Specific . . . 853
31.17. Logging Specific . . . 855
31.18. LS Specific . . . 855
31.19. GHI Specific . . . 855
31.20. FTPD Specific . . . 856
31.21. RAIT Engine Specific . . . 856
31.22. Migration Purge Server Specific . . . 856
31.23. Mover Specific . . . 857
31.24. Security Registry & Service Location Specific . . . 857
31.25. RPC Specific . . . 859
31.26. SCSI PVR . . . 859
31.27. RTM Settings . . . 860
31.28. Installation & Misc . . . 860
31.29. HPSS Group Names . . . 861
31.30. HPSS Group Default IDs . . . 861
31.31. HPSS User Names . . . 862
31.32. HPSS Server Unix UIDs . . . 862
31.33. HPSS Object Store Encryption Master Key File . . . 863
31.34. HPSS Object Store Notifcation Configuration . . . 863
Appendix E: The /var/hpss files . . . 864
Appendix F: Man Pages . . . 869
F.1. crtfileset(7) . . . 870
F.1.1. Name . . . 870
F.1.2. Syntax . . . 870
F.1.3. Parameters . . . 870
F.1.4. Description . . . 871
F.1.5. Examples . . . 871
F.1.6. See Also . . . 871
F.2. crtjunction(7) . . . 872
F.2.1. Name . . . 872
F.2.2. Syntax . . . 872
F.2.3. Description . . . 872
F.2.4. Parameters . . . 872
F.2.5. Example . . . 872
F.2.6. See Also . . . 873
F.3. deljunction(7) . . . 874
F.3.1. Name . . . 874
F.3.2. Syntax . . . 874
F.3.3. Description . . . 874
F.3.4. Parameters . . . 874
F.3.5. Example . . . 874

F.3.6. See Also . . . 875
F.4. device_scan(7) . . . 876
F.4.1. Name . . . 876
F.4.2. Syntax . . . 876
F.4.3. Description . . . 876
F.4.4. Options . . . 876
F.4.5. Examples . . . 877
F.4.6. See Also . . . 879
F.5. dump_acct_sum(7) . . . 880
F.5.1. Name . . . 880
F.5.2. Syntax . . . 880
F.5.3. Description . . . 880
F.5.4. Parameters . . . 880
F.5.5. Examples . . . 881
F.6. dumpbf(7) . . . 884
F.6.1. Name . . . 884
F.6.2. Syntax . . . 884
F.6.3. Description . . . 884
F.6.4. Options . . . 884
F.6.5. Example . . . 885
F.6.6. See Also . . . 885
F.7. dumppv_pvl(7) . . . 886
F.7.1. Name . . . 886
F.7.2. Syntax . . . 886
F.7.3. Description . . . 886
F.7.4. Options . . . 886
F.7.5. Example . . . 887
F.7.6. See Also . . . 888
F.8. dumppv_pvr(7) . . . 889
F.8.1. Name . . . 889
F.8.2. Syntax . . . 889
F.8.3. Description . . . 889
F.8.4. Options . . . 889
F.8.5. Example . . . 890
F.8.6. See Also . . . 891
F.9. dump_sspvs(7) . . . 892
F.9.1. Name . . . 892
F.9.2. Syntax . . . 892
F.9.3. Description . . . 893
F.9.4. Options . . . 893
F.9.5. Repack Output Volumes . . . 896

F.9.6. RAIT Tape Volumes . . . 896
F.9.7. Examples . . . 896
F.9.8. See Also . . . 899
F.10. ff_merge(7) . . . 900
F.10.1. Name . . . 900
F.10.2. Synopsis . . . 900
F.10.3. Description . . . 900
F.10.4. Notes . . . 900
F.10.5. Required Arguments . . . 900
F.10.6. Optional Arguments . . . 901
F.10.7. Prerequisites . . . 901
F.11. hashcheck(7) . . . 902
F.11.1. Name . . . 902
F.11.2. Syntax . . . 902
F.11.3. Description . . . 902
F.11.4. Options . . . 902
F.11.5. Examples . . . 904
F.11.6. See Also . . . 906
F.12. hpss_abort_io(7) . . . 907
F.12.1. Name . . . 907
F.12.2. Syntax . . . 907
F.12.2.1. Options for each abort type: . . . 907
F.12.3. Description . . . 907
F.12.4. Options . . . 908
F.12.5. Example . . . 908
F.13. hpss_acct(7) . . . 910
F.13.1. Name . . . 910
F.13.2. Syntax . . . 910
F.13.3. Description . . . 910
F.13.4. Parameters . . . 910
F.13.5. Example . . . 911
F.13.6. Files . . . 911
F.14. hpss_dump_acct(7) . . . 912
F.14.1. Name . . . 912
F.14.2. Syntax . . . 912
F.14.3. Description . . . 912
F.14.4. Output Format . . . 912
F.14.5. Parameters . . . 912
F.14.6. Exit Status . . . 913
F.14.7. Files . . . 913
F.14.8. Information Collected . . . 913

F.14.9. Example 1 - Display to stdout . . . 913
F.14.10. Example 2 - Log to file with ELK formatting . . . 914
F.14.11. Example 3 - Debug mode . . . 914
F.14.12. Example 4 - Custom output file . . . 914
F.14.13. See Also . . . 914
F.14.14. Notes . . . 914
F.15. hpss_dump_db_metrics(7) . . . 915
F.15.1. Name . . . 915
F.15.2. Syntax . . . 915
F.15.3. Description . . . 915
F.15.4. Output Format . . . 915
F.15.5. Parameters . . . 915
F.15.6. Exit Status . . . 916
F.15.7. Files . . . 916
F.15.8. Metrics Collected . . . 916
F.15.9. Example 1 - Display to stdout . . . 917
F.15.10. Example 2 - Log to file with ELK formatting . . . 917
F.15.11. Example 3 - Debug mode with custom subsystem . . . 917
F.15.12. Example 4 - Custom output file . . . 917
F.15.13. See Also . . . 917
F.15.14. Notes . . . 918
F.16. hpss_dump_server_metrics(7) . . . 919
F.16.1. Name . . . 919
F.16.2. Syntax . . . 919
F.16.3. Description . . . 919
F.16.4. Output Format . . . 919
F.16.5. Parameters . . . 919
F.16.6. Exit Status . . . 920
F.16.7. Files . . . 920
F.16.8. Metrics Collected . . . 920
F.16.9. Handling Unavailable Servers . . . 921
F.16.10. Example 1 - Display to stdout . . . 921
F.16.11. Example 2 - Log to file with ELK formatting . . . 922
F.16.12. Example 3 - Debug mode . . . 922
F.16.13. Example 4 - Custom output file . . . 923
F.16.14. See Also . . . 923
F.16.15. Notes . . . 923
F.17. hpssadm_devices_and_drives_list(7) . . . 924
F.17.1. Name . . . 924
F.17.2. Syntax . . . 924
F.17.3. Description . . . 924

F.17.4. Output Format . . . 924
F.17.5. Parameters . . . 924
F.17.6. Exit Status . . . 925
F.17.7. Files . . . 925
F.17.8. Example 1 - Display to stdout . . . 925
F.17.9. Example 2 - Log to file with ELK formatting . . . 926
F.17.10. Example 3 - Debug mode with custom keytab . . . 926
F.17.11. Example 4 - Custom output file . . . 927
F.17.12. See Also . . . 927
F.17.13. Notes . . . 927
F.18. hpssadm_hs_info(7) . . . 928
F.18.1. Name . . . 928
F.18.2. Syntax . . . 928
F.18.3. Description . . . 928
F.18.4. Output Format . . . 928
F.18.5. Parameters . . . 928
F.18.6. Exit Status . . . 929
F.18.7. Files . . . 929
F.18.8. Example 1 - Display to stdout . . . 929
F.18.9. Example 2 - Log to file with ELK formatting . . . 930
F.18.10. Example 3 - Debug mode with custom user . . . 930
F.18.11. See Also . . . 930
F.18.12. Notes . . . 930
F.19. hpssadm_sc_list(7) . . . 931
F.19.1. Name . . . 931
F.19.2. Syntax . . . 931
F.19.3. Description . . . 931
F.19.4. Output Format . . . 931
F.19.5. Parameters . . . 931
F.19.6. Exit Status . . . 932
F.19.7. Files . . . 932
F.19.8. Example 1 - Display to stdout . . . 932
F.19.9. Example 2 - Log to file with ELK formatting . . . 933
F.19.10. Example 3 - Debug mode with custom user . . . 933
F.19.11. Example 4 - Custom HPSS bin path . . . 933
F.19.12. See Also . . . 933
F.19.13. Notes . . . 933
F.20. hpssadm_servers_list(7) . . . 934
F.20.1. Name . . . 934
F.20.2. Syntax . . . 934
F.20.3. Description . . . 934

F.20.4. Output Format . . . 934
F.20.5. Parameters . . . 934
F.20.6. Exit Status . . . 935
F.20.7. Files . . . 935
F.20.8. Example 1 - Display to stdout . . . 935
F.20.9. Example 2 - Log to file with ELK formatting . . . 936
F.20.10. Example 3 - Debug mode with custom keytab . . . 936
F.20.11. Example 4 - Custom output file and HPSS bin path . . . 936
F.20.12. See Also . . . 936
F.20.13. Notes . . . 936
F.21. hpssadm(7) . . . 937
F.21.1. Name . . . 937
F.21.2. Table of Contents . . . 937
F.21.3. Syntax . . . 938
F.21.4. Parameters . . . 938
F.21.5. Description . . . 945
F.21.6. Environment . . . 946
F.21.7. Command Overview . . . 949
F.21.8. Structure Organization . . . 950
F.21.9. Common Command Options . . . 952
F.21.10. Specifying Multiple Command Options . . . 952
F.21.11. Specifying Fields in Batch Mode . . . 953
F.21.12. Specifying Fields in Interactive Mode . . . 958
F.21.13. Command Set . . . 960
F.21.14. Examples . . . 974
F.21.15. Files . . . 985
F.21.16. Notes . . . 987
F.22. ssmcli(7) . . . 988
F.22.1. Name . . . 988
F.22.2. Table of Contents . . . 988
F.22.3. Syntax . . . 988
F.22.4. Help . . . 988
F.22.5. Command Structure . . . 989
F.22.6. Files . . . 989
F.22.7. Description . . . 989
F.22.8. Configuration . . . 989
F.22.9. Notes . . . 990
F.23. hpss_avaledit(7) . . . 991
F.23.1. Name . . . 991
F.23.2. Syntax . . . 991
F.23.3. Description . . . 991

F.23.4. Options . . . 991
F.23.5. Example . . . 992
F.24. hpsscat(7) . . . 995
F.24.1. Name . . . 995
F.24.2. Syntax . . . 995
F.24.3. Description . . . 995
F.24.4. Options . . . 995
F.24.5. Example . . . 996
F.25. hpsschmod(7) . . . 997
F.25.1. Name . . . 997
F.25.2. Syntax . . . 997
F.25.3. Description . . . 997
F.25.4. Options . . . 997
F.25.5. Example . . . 997
F.26. hpsschown(7) . . . 999
F.26.1. Name . . . 999
F.26.2. Syntax . . . 999
F.26.3. Description . . . 999
F.26.4. Options . . . 999
F.26.5. Example . . . 999
F.27. HPSS.conf(7) . . . 1001
F.27.1. Name . . . 1001
F.27.2. Description . . . 1001
F.27.3. Location . . . 1001
F.27.4. Syntax . . . 1001
F.27.5. Examples . . . 1002
F.27.6. Files . . . 1002
F.27.7. See Also . . . 1002
F.27.8. Author . . . 1002
F.28. hpsscp(7) . . . 1003
F.28.1. Name . . . 1003
F.28.2. Synopsis . . . 1003
F.28.3. Description . . . 1003
F.28.4. Options . . . 1003
F.28.5. Example . . . 1004
F.29. hpss_create_default_vv_select_weights(7) . . . 1006
F.29.1. Name . . . 1006
F.29.2. Synopsis . . . 1006
F.29.3. Description . . . 1006
F.29.4. Required Arguments . . . 1007
F.29.5. Optional Arguments . . . 1007

F.29.6. Prerequisites . . . 1007
F.30. hpss_db_metrics(7) . . . 1008
F.30.1. Name . . . 1008
F.30.2. Syntax . . . 1008
F.30.3. Description . . . 1008
F.30.4. Parameters . . . 1008
F.30.5. Examples . . . 1009
F.31. hpss_db_diag(7) . . . 1012
F.31.1. Name . . . 1012
F.31.2. Syntax . . . 1012
F.31.3. Description . . . 1012
F.31.4. Parameters . . . 1012
F.31.5. Examples . . . 1013
F.32. hpss_delog(7) . . . 1015
F.32.1. Name . . . 1015
F.32.2. Syntax . . . 1015
F.32.3. Description . . . 1015
F.32.4. Arguments . . . 1016
F.32.5. Examples . . . 1017
F.32.6. Limitations . . . 1017
F.33. hpss_drive_ctl(7) . . . 1018
F.33.1. Name . . . 1018
F.33.2. Syntax . . . 1018
F.33.3. Description . . . 1018
F.33.4. Options . . . 1018
F.33.5. Examples . . . 1019
F.34. hpss_errno(7) . . . 1020
F.34.1. Name . . . 1020
F.34.2. Syntax . . . 1020
F.34.3. Description . . . 1020
F.34.4. Options . . . 1020
F.34.5. Examples . . . 1020
F.35. hpssfs(7) . . . 1021
F.35.1. NAME . . . 1021
F.35.2. SYNOPSIS . . . 1021
F.35.3. DESCRIPTION . . . 1021
F.35.4. OPTIONS . . . 1021
F.35.4.1. hpssfs-specific options . . . 1021
F.35.4.2. Supported FUSE mount options . . . 1024
F.35.4.3. Supported kernel mount options . . . 1024
F.35.5. EXAMPLES . . . 1025

F.35.6. SEE ALSO . . . 1026
F.36. hpssfs_undelete(7) . . . 1027
F.36.1. NAME . . . 1027
F.36.2. SYNOPSIS . . . 1027
F.36.3. DESCRIPTION . . . 1027
F.36.4. OPTIONS . . . 1027
F.36.5. EXAMPLES . . . 1027
F.36.6. NOTES . . . 1027
F.36.7. SEE ALSO . . . 1028
F.37. hpss_ftp(7) . . . 1029
F.37.1. Name . . . 1029
F.37.2. Description . . . 1029
F.37.3. Building a Configuration File . . . 1029
F.37.4. Files . . . 1030
F.37.5. See Also . . . 1030
F.37.6. Author . . . 1030
F.38. hpssgetvol(7) . . . 1031
F.38.1. Name . . . 1031
F.38.2. Syntax . . . 1031
F.38.3. Description . . . 1031
F.38.4. Options . . . 1031
F.38.5. Example . . . 1031
F.39. hpssgetxattrs(7) . . . 1033
F.39.1. Name . . . 1033
F.39.2. Syntax . . . 1033
F.39.3. Description . . . 1033
F.39.4. Example . . . 1034
F.40. hpssgui(7) . . . 1037
F.40.1. Name . . . 1037
F.40.2. Syntax . . . 1037
F.40.3. Description . . . 1045
F.40.4. Environment . . . 1046
F.40.5. Files . . . 1049
F.40.6. Notes . . . 1051
F.40.7. Related Information . . . 1053
F.41. hpsslimits(7) . . . 1054
F.41.1. Name . . . 1054
F.41.2. Syntax . . . 1054
F.41.3. Description . . . 1054
F.41.4. Options . . . 1054
F.41.5. Example . . . 1055

F.42. hpssln(7) . . . 1056
F.42.1. Name . . . 1056
F.42.2. Syntax . . . 1056
F.42.3. Description . . . 1056
F.42.4. Options . . . 1056
F.42.5. Example . . . 1057
F.43. hpss_license(7) . . . 1058
F.43.1. Name . . . 1058
F.43.2. Syntax . . . 1058
F.43.3. Description . . . 1058
F.43.4. Parameters . . . 1058
F.43.5. Examples . . . 1058
F.44. hpss_log_archive(7) . . . 1059
F.44.1. Name . . . 1059
F.44.2. Syntax . . . 1059
F.44.3. Description . . . 1059
F.44.4. Options . . . 1060
F.44.5. Example . . . 1061
F.45. hpss_log_formatter(7) . . . 1062
F.45.1. Name . . . 1062
F.45.2. Syntax . . . 1062
F.45.3. Description . . . 1062
F.45.4. Options . . . 1062
F.45.5. Examples . . . 1062
F.46. hpssls(7) . . . 1065
F.46.1. Name . . . 1065
F.46.2. Syntax . . . 1065
F.46.3. Description . . . 1065
F.46.4. Options . . . 1065
F.46.5. Example . . . 1066
F.47. hpssmkdir(7) . . . 1067
F.47.1. Name . . . 1067
F.47.2. Syntax . . . 1067
F.47.3. Description . . . 1067
F.47.4. Options . . . 1067
F.47.5. Example . . . 1068
F.48. hpssmsg(7) . . . 1069
F.48.1. Name . . . 1069
F.48.2. Syntax . . . 1069
F.48.3. Description . . . 1069
F.48.4. Parameters . . . 1069

F.48.5. Examples . . . 1070
F.48.6. See Also . . . 1070
F.49. hpss_pftpd(7) . . . 1071
F.49.1. Name . . . 1071
F.49.2. Synopsis . . . 1071
F.49.3. Description . . . 1071
F.49.4. Options . . . 1071
F.49.5. Supported File Transfer Protocol Requests . . . 1071
F.49.6. HPSS Supported FTP Extensions . . . 1074
F.49.7. Building a Configuration File . . . 1076
F.49.8. Files . . . 1076
F.49.9. Environment . . . 1076
F.49.10. Examples . . . 1076
F.49.11. See Also . . . 1077
F.49.12. Author . . . 1077
F.50. hpsspurge(7) . . . 1078
F.50.1. Name . . . 1078
F.50.2. Syntax . . . 1078
F.50.3. Description . . . 1078
F.50.4. Options . . . 1078
F.50.5. Example . . . 1078
F.51. hpsspurgelock(7) . . . 1080
F.51.1. Name . . . 1080
F.51.2. Syntax . . . 1080
F.51.3. Description . . . 1080
F.51.4. Options . . . 1080
F.51.5. Example . . . 1081
F.52. hpss_recurse(7) . . . 1082
F.52.1. Name . . . 1082
F.52.2. Syntax . . . 1082
F.52.3. Description . . . 1082
F.52.4. Options . . . 1082
F.52.5. Examples . . . 1083
F.53. hpssrm(7) . . . 1087
F.53.1. Name . . . 1087
F.53.2. Syntax . . . 1087
F.53.3. Description . . . 1087
F.53.4. Options . . . 1087
F.53.5. Example . . . 1087
F.54. hpssrmdir(7) . . . 1089
F.54.1. Name . . . 1089

F.54.2. Syntax . . . 1089
F.54.3. Description . . . 1089
F.54.4. Options . . . 1089
F.54.5. Example . . . 1089
F.55. hpss_server_acl(7) . . . 1091
F.55.1. Name . . . 1091
F.55.2. Syntax . . . 1091
F.55.3. Description . . . 1091
F.55.4. Commands . . . 1091
F.55.5. ACL Syntax . . . 1092
F.55.6. Interface Selection . . . 1093
F.55.7. Examples . . . 1093
F.56. hpss_server_metrics(7) . . . 1095
F.56.1. Name . . . 1095
F.56.2. Syntax . . . 1095
F.56.3. Description . . . 1095
F.56.4. Parameters . . . 1096
F.56.5. Handling Unavailable Servers . . . 1096
F.56.5.1. Behavior . . . 1096
F.56.6. Example 1 . . . 1097
F.56.7. Example 2 - Unavailable Mover . . . 1097
F.57. hpsssetcos(7) . . . 1099
F.57.1. Name . . . 1099
F.57.2. Syntax . . . 1099
F.57.3. Description . . . 1099
F.57.4. Options . . . 1099
F.57.5. Example . . . 1100
F.58. hpss_set_env(7) . . . 1101
F.58.1. Name . . . 1101
F.58.2. Syntax . . . 1101
F.58.3. Description . . . 1101
F.58.4. Examples . . . 1101
F.58.5. Files . . . 1102
F.58.6. See Also . . . 1102
F.59. hpssstat(7) . . . 1103
F.59.1. Name . . . 1103
F.59.2. Syntax . . . 1103
F.59.3. Description . . . 1103
F.59.4. Options . . . 1103
F.59.5. Example . . . 1103
F.60. hpsssum(7) . . . 1105

F.60.1. Name . . . 1105
F.60.2. Syntax . . . 1105
F.60.3. Description . . . 1105
F.60.4. Options . . . 1105
F.60.5. Examples . . . 1108
F.60.6. See Also . . . 1111
F.61. hpsstouch(7) . . . 1112
F.61.1. Name . . . 1112
F.61.2. Syntax . . . 1112
F.61.3. Description . . . 1112
F.61.4. Options . . . 1112
F.61.5. Example . . . 1112
F.62. hpss_san3p_part(7) . . . 1114
F.62.1. Name . . . 1114
F.62.2. Syntax . . . 1114
F.62.3. Description . . . 1114
F.62.4. Parameters . . . 1114
F.62.5. Examples . . . 1115
F.63. hpss_unix_auth(7) . . . 1116
F.63.1. Name . . . 1116
F.63.2. Syntax . . . 1116
F.63.3. Description . . . 1116
F.63.4. Options . . . 1116
F.63.5. Examples . . . 1117
F.64. hpss_unix_group(7) . . . 1118
F.64.1. Name . . . 1118
F.64.2. Syntax . . . 1118
F.64.3. Description . . . 1118
F.64.4. Options . . . 1118
F.64.5. Examples . . . 1119
F.65. hpss_unix_keytab(7) . . . 1120
F.65.1. Name . . . 1120
F.65.2. Syntax . . . 1120
F.65.3. Description . . . 1120
F.65.4. Options . . . 1120
F.65.5. Examples . . . 1122
F.65.6. Files . . . 1122
F.65.7. See Also . . . 1122
F.66. hpss_unix_passwd(7) . . . 1123
F.66.1. Name . . . 1123
F.66.2. Syntax . . . 1123

F.66.3. Description . . . 1123
F.66.4. Options . . . 1123
F.67. hpss_unix_user(7) . . . 1124
F.67.1. Name . . . 1124
F.67.2. Syntax . . . 1124
F.67.3. Description . . . 1124
F.67.4. Options . . . 1124
F.67.5. Examples . . . 1125
F.68. hpss_update_priority_by_create_time(7) . . . 1126
F.68.1. Name . . . 1126
F.68.2. Synopsis . . . 1126
F.68.3. Description . . . 1126
F.68.4. Required Arguments . . . 1126
F.68.5. Optional Arguments . . . 1126
F.68.6. Prerequisites . . . 1126
F.69. hpss_update_priority_by_pvr(7) . . . 1127
F.69.1. Name . . . 1127
F.69.2. Synopsis . . . 1127
F.69.3. Description . . . 1127
F.69.4. Configuration . . . 1127
F.69.5. Required Arguments . . . 1128
F.69.6. Optional Arguments . . . 1128
F.69.7. Prerequisites . . . 1128
F.70. hpss_update_priority_by_pvr_drive(7) . . . 1129
F.70.1. Name . . . 1129
F.70.2. Synopsis . . . 1129
F.70.3. Description . . . 1129
F.70.4. Configuration . . . 1129
F.70.5. Required Arguments . . . 1130
F.70.6. Option Arguments . . . 1130
F.70.7. Prerequisites . . . 1130
F.71. hpssuser(7) . . . 1131
F.71.1. Name . . . 1131
F.71.2. Syntax . . . 1131
F.71.3. Description . . . 1131
F.71.4. Actions . . . 1131
F.71.5. User Types . . . 1132
F.71.6. Options . . . 1132
F.71.7. Example . . . 1133
F.71.8. Notes . . . 1134
F.71.9. Environment . . . 1134

F.71.10. Files . . . 1135
F.71.11. See Also . . . 1135
F.72. hpss_vv_select_priority(7) . . . 1136
F.72.1. Name . . . 1136
F.72.2. Synopsis . . . 1136
F.72.3. Description . . . 1136
F.72.4. Required Arguments . . . 1136
F.72.5. Optional Arguments . . . 1137
F.72.6. Prerequisites . . . 1137
F.73. hpssxfind(7) . . . 1138
F.73.1. Name . . . 1138
F.73.2. Syntax . . . 1138
F.73.3. Description . . . 1138
F.73.4. Options . . . 1138
F.73.5. Example . . . 1139
F.74. lbp_verify(7) . . . 1141
F.74.1. Name . . . 1141
F.74.2. Syntax . . . 1141
F.74.3. Description . . . 1141
F.74.4. Options . . . 1141
F.74.5. RAIT Tape Volumes . . . 1144
F.74.6. Admin Abort . . . 1144
F.74.7. Premature Verify Halt . . . 1144
F.74.8. Examples . . . 1145
F.74.9. See Also . . . 1146
F.75. lori(7) . . . 1147
F.75.1. Name . . . 1147
F.75.2. Syntax . . . 1147
F.75.3. Description . . . 1147
F.75.4. Input Methods . . . 1147
F.75.5. Options . . . 1148
F.75.6. Examples . . . 1151
F.75.7. Output Format . . . 1151
F.75.8. Filters . . . 1152
F.76. lscos(7) . . . 1154
F.76.1. Name . . . 1154
F.76.2. Syntax . . . 1154
F.76.3. Description . . . 1154
F.76.4. Parameters . . . 1154
F.76.5. Example 1 . . . 1155
F.76.6. Example 2 . . . 1155

F.76.7. Example 3 . . . 1156
F.76.8. Example 4 . . . 1156
F.76.9. Note . . . 1157
F.77. lsfilesets(7) . . . 1158
F.77.1. Name . . . 1158
F.77.2. Syntax . . . 1158
F.77.3. Description . . . 1158
F.77.4. Parameters . . . 1158
F.77.5. Example . . . 1158
F.78. lshpss(7) . . . 1161
F.78.1. Name . . . 1161
F.78.2. Syntax . . . 1161
F.78.3. Description . . . 1161
F.78.4. Options . . . 1161
F.78.5. Example . . . 1163
F.79. lsjunctions(7) . . . 1164
F.79.1. Name . . . 1164
F.79.2. Syntax . . . 1164
F.79.3. Description . . . 1164
F.79.4. Parameters . . . 1164
F.79.5. Example . . . 1164
F.79.6. See Also . . . 1165
F.80. lspvhist(7) . . . 1166
F.80.1. Name . . . 1166
F.80.2. Syntax . . . 1166
F.80.3. Description . . . 1166
F.80.4. Options . . . 1167
F.80.5. Examples . . . 1169
F.81. lsrb(7) . . . 1172
F.81.1. Name . . . 1172
F.81.2. Syntax . . . 1172
F.81.3. Description . . . 1172
F.81.4. Options . . . 1172
F.81.5. Example . . . 1173
F.81.6. See Also . . . 1173
F.82. lsvol(7) . . . 1174
F.82.1. Name . . . 1174
F.82.2. Syntax . . . 1174
F.82.3. Description . . . 1174
F.82.4. Options . . . 1175
F.82.5. Example . . . 1176

F.82.6. See Also . . . 1177
F.83. lsvol_graph(7) . . . 1178
F.83.1. Name . . . 1178
F.83.2. Syntax . . . 1178
F.83.3. Prerequisites . . . 1178
F.83.4. Description . . . 1178
F.83.5. A caveat on tape marks . . . 1178
F.83.6. Options . . . 1179
F.83.7. Example . . . 1180
F.83.8. See Also . . . 1180
F.84. manage_temporal(7) . . . 1181
F.84.1. Name . . . 1181
F.84.2. Synopsis . . . 1181
F.84.3. Description . . . 1181
F.84.4. Options . . . 1181
F.84.5. Initial Installation . . . 1183
F.84.5.1. Installing the Python Database Interface Package . . . 1183
F.84.5.2. Adding libdb2.so.1 to the Library Search Path . . . 1183
F.84.5.3. Installing the UDF Routines . . . 1183
F.84.6. Caveats . . . 1183
F.84.7. Author . . . 1184
F.85. hpss_add_temporal_tables(7) . . . 1185
F.85.1. Name . . . 1185
F.85.2. Synopsis . . . 1185
F.85.3. Description . . . 1185
F.85.4. Options . . . 1185
F.86. hpss_remove_temporal_tables(7) . . . 1187
F.86.1. Name . . . 1187
F.86.2. Synopsis . . . 1187
F.86.3. Description . . . 1187
F.86.4. Options . . . 1187
F.87. mkmprec(7) . . . 1188
F.87.1. Name . . . 1188
F.87.2. Syntax . . . 1188
F.87.3. Description . . . 1188
F.87.4. Options . . . 1188
F.87.5. Example . . . 1189
F.88. mps_reporter(7) . . . 1190
F.88.1. Name . . . 1190
F.88.2. Syntax . . . 1190
F.88.3. Description . . . 1190

F.88.4. Options . . . 1190
F.88.5. Notes . . . 1191
F.88.6. Examples . . . 1191
F.89. mps_reporter_legacy(7) . . . 1194
F.89.1. Name . . . 1194
F.89.2. Syntax . . . 1194
F.89.3. Description . . . 1194
F.89.4. Options . . . 1194
F.89.5. Notes . . . 1194
F.89.6. Examples . . . 1195
F.90. multinoded(7) . . . 1198
F.90.1. Name . . . 1198
F.90.2. Synopsis . . . 1198
F.90.3. Description . . . 1198
F.90.4. Options . . . 1198
F.90.5. Configure a Node to Run the HPSS Multinode Daemon . . . 1198
F.90.6. Building a Configuration File . . . 1199
F.90.7. Files . . . 1199
F.90.8. See Also . . . 1199
F.90.9. Author . . . 1199
F.91. pftp_client(7) . . . 1200
F.91.1. Name . . . 1200
F.91.2. Synopsis . . . 1200
F.91.3. Description . . . 1200
F.91.4. Options . . . 1200
F.91.5. Issuing Subcommands . . . 1202
F.91.6. Standard FTP Compatible Subcommands . . . 1202
F.91.7. Unsupported FTP Subcommands . . . 1202
F.91.8. Extended PFTP Subcommands . . . 1203
F.91.9. Building a Configuration File . . . 1206
F.91.10. Files . . . 1206
F.91.11. Environment . . . 1206
F.91.12. See Also . . . 1206
F.91.13. Author . . . 1206
F.92. plu(7) . . . 1207
F.92.1. Name . . . 1207
F.92.2. Syntax . . . 1207
F.92.3. Description . . . 1207
F.92.4. Options . . . 1207
F.92.5. Examples . . . 1210
F.93. quaid(7) . . . 1212

F.93.1. Name . . . 1212
F.93.2. Syntax . . . 1212
F.93.3. Description . . . 1212
F.93.4. Input Methods . . . 1213
F.93.5. Options . . . 1213
F.93.6. Note . . . 1215
F.93.7. Examples . . . 1215
F.93.8. Output Format . . . 1216
F.93.9. Filters . . . 1217
F.93.9.1. Limitations and Performance . . . 1218
F.94. rc.hpss(7) . . . 1219
F.94.1. Name . . . 1219
F.94.2. Syntax . . . 1219
F.94.3. Description . . . 1219
F.94.4. Options . . . 1219
F.94.5. Examples . . . 1219
F.95. reclaim(7) . . . 1220
F.95.1. Name . . . 1220
F.95.2. Syntax . . . 1220
F.95.3. Description . . . 1220
F.95.4. Options . . . 1221
F.95.5. Examples . . . 1222
F.95.6. See Also . . . 1223
F.96. recover(7) . . . 1224
F.96.1. Name . . . 1224
F.96.2. Syntax . . . 1224
F.96.2.1. Command line form 1: . . . 1224
F.96.2.2. Command line form 2: . . . 1224
F.96.2.3. Command line form 3: . . . 1224
F.96.2.4. Command line form 4: . . . 1225
F.96.2.5. Command line form 5: . . . 1225
F.96.3. Description . . . 1226
F.96.4. Options . . . 1228
F.96.5. Alternate Copies . . . 1230
F.96.6. Summary Status . . . 1231
F.96.7. Report File . . . 1234
F.96.8. Examples . . . 1237
F.96.8.1. Run recover in recovery mode . . . 1237
F.96.8.2. Run recover in cleanup mode . . . 1238
F.96.8.3. List volumes currently undergoing recovery or cleanup . . . 1238
F.96.8.4. Display the status of a volume currently undergoing recovery or cleanup. . . . 1239

F.96.9. Notes . . . 1240
F.96.10. Files . . . 1241
F.96.11. See Also . . . 1241
F.97. rehome_files(7) . . . 1242
F.97.1. Name . . . 1242
F.97.2. Syntax . . . 1242
F.97.3. Description . . . 1242
F.97.4. Options . . . 1243
F.97.5. Notes . . . 1244
F.97.6. Examples . . . 1245
F.97.7. See Also . . . 1245
F.98. remove(7) . . . 1246
F.98.1. Name . . . 1246
F.98.2. Syntax . . . 1246
F.98.3. Description . . . 1246
F.98.4. Options . . . 1247
F.98.5. Example . . . 1248
F.98.6. See Also . . . 1249
F.99. repack(7) . . . 1250
F.99.1. Name . . . 1250
F.99.2. Syntax . . . 1250
F.99.3. Description . . . 1252
F.99.4. Options . . . 1253
F.99.5. Selection Criteria . . . 1256
F.99.6. Environment . . . 1257
F.99.7. Notes . . . 1257
F.99.8. Example . . . 1257
F.99.9. See Also . . . 1259
F.100. retire(7) . . . 1260
F.100.1. Name . . . 1260
F.100.2. Syntax . . . 1260
F.100.3. Description . . . 1260
F.100.4. Options . . . 1261
F.100.5. Example . . . 1262
F.100.6. See Also . . . 1262
F.101. rq_cleanup(7) . . . 1263
F.101.1. Name . . . 1263
F.101.2. Synopsis . . . 1263
F.101.3. Description . . . 1263
F.101.4. Notes . . . 1263
F.101.5. Required Arguments . . . 1263

F.101.6. Optional Arguments . . . 1263
F.101.7. Prerequisites . . . 1264
F.102. rtmu(7) . . . 1265
F.102.1. Name . . . 1265
F.102.2. Syntax . . . 1265
F.102.3. Description . . . 1265
F.102.4. Options . . . 1267
F.102.5. Examples . . . 1268
F.102.6. Short Format Field Definitions . . . 1269
F.102.7. Medium Format Abbreviations . . . 1270
F.102.8. Long Format Field Definitions . . . 1272
F.102.8.1. Action Filters . . . 1276
F.103. schema_registrar(7) . . . 1277
F.103.1. Name . . . 1277
F.103.2. Syntax . . . 1277
F.103.3. Description . . . 1277
F.103.4. Options . . . 1277
F.103.5. Examples . . . 1277
F.104. scrub(7) . . . 1278
F.104.1. Name . . . 1278
F.104.2. Syntax . . . 1278
F.104.3. Description . . . 1278
F.104.4. Commands . . . 1278
F.104.5. Usage . . . 1278
F.104.6. Example . . . 1278
F.104.7. Files . . . 1279
F.105. scsi_home(7) . . . 1280
F.105.1. Name . . . 1280
F.105.2. Syntax . . . 1280
F.105.3. Description . . . 1280
F.105.4. Options . . . 1281
F.105.5. Examples . . . 1281
F.106. shelf_tape(7) . . . 1284
F.106.1. Name . . . 1284
F.106.2. Syntax . . . 1284
F.106.3. Description . . . 1284
F.106.3.1. Tape Checkout Requests . . . 1284
F.106.3.2. Tape Checkin Requests . . . 1284
F.106.4. Options . . . 1284
F.106.5. Example . . . 1286
F.106.6. See Also . . . 1287

F.107. shelf_tape_util(7) . . . 1288
F.107.1. Name . . . 1288
F.107.2. Syntax . . . 1288
F.107.3. Description . . . 1288
F.107.4. Options . . . 1288
F.107.5. Example . . . 1289
F.107.6. See Also . . . 1289
F.108. site.conf(7) . . . 1290
F.108.1. Name . . . 1290
F.108.2. Description . . . 1290
F.108.3. Location . . . 1290
F.108.4. Syntax . . . 1290
F.108.4.1. Site Name . . . 1290
F.108.4.2. Global Settings . . . 1290
F.108.4.3. Realm definition . . . 1291
F.108.5. Example . . . 1293
F.108.6. See Also . . . 1294
F.108.7. Author . . . 1294
F.109. ssmapi_token(7) . . . 1295
F.109.1. Name . . . 1295
F.109.2. Synopsis . . . 1295
F.109.3. Description . . . 1295
F.109.4. Required Arguments . . . 1295
F.109.5. Optional Arguments . . . 1295
F.109.6. Prerequisites . . . 1295
F.110. xmladdindex(7) . . . 1296
F.110.1. Name . . . 1296
F.110.2. Syntax . . . 1296
F.110.3. Description . . . 1296
F.110.4. Options . . . 1296
F.110.5. Examples . . . 1296
F.110.6. See Also . . . 1297
F.111. xmldeleteindex(7) . . . 1298
F.111.1. Name . . . 1298
F.111.2. Syntax . . . 1298
F.111.3. Description . . . 1298
F.111.4. Options . . . 1298
F.111.5. Examples . . . 1298
F.111.6. See Also . . . 1298

Copyright notification

Copyright © 1992-2026 International Business Machines Corporation, The Regents of the University of California, Triad National Security, LLC, Lawrence Livermore National Security, LLC, National Technology & Engineering Solutions of Sandia, LLC, and UT-Battelle.

All rights reserved.

Portions of this work were produced by Lawrence Livermore National Security, LLC, Lawrence Livermore National Laboratory (LLNL) under Contract No. DE-AC52-07NA27344 with the U.S. Department of Energy (DOE); by the University of California, Lawrence Berkeley National Laboratory (LBNL) under Contract No. DE-AC02-05CH11231 with DOE; by Triad National Security, LLC, Los Alamos National Laboratory (LANL) under Contract No. 89233218CNA000001 with DOE; by National Technology & Engineering Solutions of Sandia, LLC (NTESS), Sandia National Laboratories (SNL) under Contract No. DE-NA0003525 with DOE; and by UT-Battelle, Oak Ridge National Laboratory (ORNL) under Contract No. DE-AC05-00OR22725 with DOE. The U.S. Government has certain reserved rights under its prime contracts with the Laboratories.

DISCLAIMER

Portions of this software were sponsored by an agency of the United States Government. Neither the United States, DOE, The Regents of the University of California, Triad National Security, LLC, Lawrence Livermore National Security, LLC, National Technology & Engineering Solutions of Sandia, LLC, UT-Battelle, nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

Trademark usage

High Performance Storage System is a trademark of International Business Machines Corporation.

IBM is a registered trademark of International Business Machines Corporation.

IBM, DB2, DB2 Universal Database, pSeries, and xSeries are trademarks or registered trademarks of International Business Machines Corporation.

UNIX is a registered trademark of the Open Group.

Linux is a registered trademark of Linus Torvalds in the United States and other countries.

Kerberos is a trademark of the Massachusetts Institute of Technology.

Java is a registered trademark of Oracle and/or its affiliates.

ACSLS is a trademark of Oracle and/or its affiliates.

Microsoft Windows is a registered trademark of Microsoft Corporation.

Other brands and product names appearing herein may be trademarks or registered trademarks of third parties.

About this book

The HPSS Admin Guide is for use both at system installation time as well as throughout the lifetime of the system. It will guide system administrators through the planning, installation, and administration of a new HPSS system. It serves as a reference whenever the system is reconfigured by the addition, deletion, or modification of hosts, tape libraries, devices, or other components.

Conventions used in this book

Example commands that should be typed at a command line will be proceeded by a percent sign ("%") and be presented in a courier font:

```
% sample command
```

Example command output and example contents of ASCII files will be presented in a courier font:

```
sample file line 1

sample file line 2
```

Any text preceded by a pound sign ("#") should be considered comment lines:

```
# This is a comment
```

Chapter 1. Release 12

This chapter summarizes HPSS changes for Release 12 into four categories: new features, retired features, deprecated features, and changed features. Changes since release 11.3 are described. For changes prior to 12.1, consult prior release notes and the HPSS features section.

1.1. New features in 12.1

New features for 12.1 are described below.

1.1.1. SSM Command Line Interface (SSM CLI)

HPSS now provides a command-line interface (CLI) for the SSM REST API. The SSM CLI allows administrators and operators to interact with SSM directly from the command line, providing a simple and convenient way to perform management tasks. It also simplifies automation and batch workflows without requiring direct REST API requests.

Some key features:

- Supports JSON, YAML, and other structured output formats for easy parsing and automation.
- Reduced startup time and lower command latency compared to ADM—ideal for scripts, pipelines, and high-frequency operational tasks.
- Portable across environments; runs anywhere the SSM REST API is reachable (e.g., Linux, macOS, Windows) with no dependency on platform-specific HPSS binaries.
- Modern CLI design with descriptive help text and consistent subcommands, making it more intuitive and easier to use than ADM.

The SSM CLI is intended to replace the legacy hpssadm interface, providing a modern, REST-based alternative for administrative and operational workflows. The CLI is currently under active development and will continue to be expanded until it fully replaces ADM functionality.

See the Quick startup of the SSM CLI for more details.

1.1.2. Limited Life-Time Session Tokens for S3 Proxy

S3 Proxy now provides support for AWS-like session tokens for end user authentication. The session token mechanism is implemented to work with Keycloak compliant OpenID Connect (OIDC) providers. See the **Setting up for limited lifetime session tokens** section in the HPSS S3 Interface Guide for more details.

1.1.3. Support for LTO-10 Enterprise Media

Support added for LTO-10 Enterprise media.

1.1.4. Encrypted Data Transfers

HPSS now supports automatic data encryption during transfers. Encryption is generally enabled by configuring HPSS_API_ENCRYPT=1 in `env.conf` on the client. HPSS will generate an ephemeral key and negotiate the encryption as part of the transfer request.

Data encryption is disabled by default.

Note that this is distinct from encrypting RPC control data (see Security controls).

Client applications may present options to enable encryption differently - consult man pages or usage for specific information.

The following supported HPSS client applications do not yet support encryption:

- GHI
- HSI
- HTAR

See the HPSS Programmer's Reference for more information about enabling encryption in other HPSS client applications.

1.1.5. License Key for HPSS Core Servers

HPSS Core Servers now require an installed, limited time license key. The license keys must be installed in the HPSS_PATH_ETC directory (for example, /var/hpss/etc). HPSS support will provide you with license keys for each HPSS instance.

WARN: The license keys must be installed before HPSS is started, or else HPSS will need to be restarted.

See Install HPSS License for more details.

1.1.6. Mover control channel encryption

By specifying an encryption algorithm for the mover in site.conf, mover control communications will be encrypted. See the man page for `site.conf` for details.

1.1.7. Port Range for Client API Applications

There is a new environment variable, HPSS_API_PORT_RANGE, the HPSS Client API can use.

See Client API configuration for more details.

1.2. Features retired in 12.1

None.

1.3. Features deprecated in 12.1

None.

1.4. Changes to existing HPSS features in 12.1

1.4.1. Updates to the SSM API

The SSM API now provides additional functionality through following new endpoints:

- **/references/errors/** - Provides the error code and message based on the HPSS error number.

1.5. API changes in HPSS 12.1

1.5.1. Changes to PIO

This applies to the advanced PIO interface only. hpss_PIOStart now requires that the `hpss_pio_params_t` structure be initialized to zero prior to its members being set or being passed in. Failure to do this will cause the hpss_PIOImportGrp call to fail with an -22 (EINVAL).

Example:

```
hpss_pio_params_t params;
params.Operation = HPSS_PIO_READ;
rc = hpss_PIOStart(&params, &grp);    // Bad!

hpss_pio_params_t params
memset(&params, 0x0, sizeof(params));
params.Operation = HPSS_PIO_READ;
rc = hpss_PIOStart(&params, &grp);    // Good!
```

1.5.2. HPSS Client API Python Binding Changes (hpsspy)

- The hpss.StageCallBackBitfile previously returned the RequestID as a tuple (e.g. (reqid)), and now returns the RequestID.

Example:

```
Old:
(reqid,) = hpss.StageCallBackBitfile(...)

New:
reqid = hpss.StageCallBackBitfile(...)
```

Chapter 2. Release 11

This chapter summarizes HPSS changes for Release 11 into four categories: new features, retired features, deprecated features, and changed features. Changes since release 10.3 are described. For changes prior to 11.1, consult prior release notes and the HPSS features section.

2.1. New features in 11.3

New features for 11.3 are described below.

2.1.1. Mark PVR Disabled

PVRs can now be marked as disabled in their configuration. This state persists whether the PVR is running or not. This flag should be set when the library is down for maintenance or the PVR is otherwise not usable.

In general, marking a PVR as disabled will cause HPSS to avoid using that library. One example is that all volumes are treated by the core server as if they were DOWN.

See Taking a tape library down for maintenance for more information and details.

2.1.2. Persistent Asynchronous Stage Requests

HPSS now automatically persists stages made with the stage callback API using the core server request queue mechanisms. Read and stage requests are optimized together using RAO in this queue, with persistence and server memory-limited capacity. Stage requests which were issued using these APIs will now resume after a core server restart. Quaid is now using this mechanism for its batch stage operations. See the Overview of TOR section for details.

HPSS now provides the capability to issue asynchronous stage requests without stage completion callbacks. This is for applications that want to issue an asynchronous stage, but do not want to receive a callback about its completion. This capability is enabled using the **BFS_NO_CB** flag for the *hpss_StageCallback*()* APIs (see the **HPSS Programmers Reference** for details).

2.1.3. UNIX Keytab Shadow File

Keytabs were originally based on the user's shadow entry. However, modern UNIX systems may use different mechanisms that do not even involve a shadow entry. Examples include LDAP and ActiveDirectory-based logins. While it is possible to use passwords by making the user enter them, this is impractical for keytabs. Therefore the decision was made to separate the two both in function and in implementation.

If using UNIX authentication, keytabs and login passwords now use separate files. There is a new file which exists in the same directory as the HPSS Shadow file, which takes the name of password shadow file with a prefix of "kt." So the default name is "/var/hpss/etc/kt.shadow". This file will maintain the list of keytab keys.

The file must exist for the HPSS system to work. This also means that changing a system password will not affect the keytab secret, or vice versa.

The kt.shadow file will be created and populated by the tools/convert/hpss/ConvertSec script, but it will only have entries which are in the hpss.unix.keytab file. To create the kt.shadow file manually, there are some options, based on your UNIX configuration.

For HPSS UNIX, simply copying /var/hpss/etc/shadow to /var/hpss/etc/kt.shadow will be sufficient to upgrade the system. Note that while the two files will begin with identical contents, they will diverge as passwords and keytabs are modified, created, or destroyed.

For system UNIX, if you are using a local shadow file, that file can be copied to /var/hpss/etc/kt.shadow. It may be worthwhile, though, to cull the kt.shadow file of accounts that are not useful for HPSS, such as *bin*, *adm*, daemon accounts, and users who do not actually have HPSS access.

If using system UNIX and an external authentication service, it will be necessary to create the kt.shadow file as an empty file, then fill it anew. This can be done by listing the current keytab entries in /var/hpss/etc/hpss.unix.keytab and then running hpss_unix_keytab -S -r update <entry> to create a new keytab secret for each entry and update the keytab with that secret. It is also worth mentioning that this method MAY be used with HPSS UNIX also, if it is desirable to decouple login passwords from keytabs immediately.

```
# touch /var/hpss/etc/kt.shadow
# chmod 0600 /var/hpss/etc/kt.shadow
# for u in `/opt/hpss/bin/hpss_unix_keytab list`; do
    /opt/hpss/bin/hpss_unix_keytab -S -r update $u
  done
```

2.2. HPSS Config History

HPSS now supports retrieving the history of configuration changes. This feature can help an administrator understand changes to the system configuration over time, and when changes were applied to the system.

The primary means of interacting with HPSS config history is through the new `hpss_cfg_history` tool. This tool displays the changes to the system from a given timeframe in JSON format. As an example, this will show the changes since October 30, 2024 at 16:32:11.

```
%/opt/hpss/bin/hpss_cfg_history -a -T 2024-10-30-16.32.11
{"since": "2024-10-30-16.32.11", "core": {"5": {"MAX_ACTIVE_IO_REQS": {"old": 4990,
"new": 100}}}, "server": {"Core Server": {"REQUEST_QUEUE_SIZE": {"old": 2500, "new":
1000}, "THREAD_POOL_SIZE": {"old": 5000, "new": 1000}, "MAX_CONNECTIONS": {"old":
5000, "new": 2000}}}}
```

The output here has the type (core), followed by the ID (server ID 5), followed by the fields that have been changed. Note that some information (e.g. core server changes) are split between two

definitions - one is the basic server configuration under "Core" and the other is under "Server". The output shows the old and new values. By default, if no time is provided with -T, the tool will show changes in the last 24 hours.

In the example, changes for all tables are shown, which is done via the "-a" option. If an administrator just wanted to see the changes to the server table, he could replace "-a" with "-l server". If he just wanted to see the server and core tables, he could instead use "-l server,core".

See the `hpss_cfg_history` page for more details.

2.2.1. New S3 Proxy Headers and Configuration

S3 Proxy now provides support for setting the client stripe transfer width via an http header (*x-amz-meta-hpss-swidth*) or proxy configuration property (*jclouds.pftp.hpss.swidth*). Client stripe transfers can potentially increase transfer performance. See the **HPSS S3 Interface Setup** and **Specific HPSS Options** sections in the HPSS S3 Interface Guide for more details.

S3 Proxy now supports an http header (*x-amz-meta-hpss-mpp-length*) that allows the user to provide the total multipart upload size to aid HPSS in segment sizing for classes of service configured to use fixed/classic style allocation, which can reduce metadata overhead and increase performance. See the **Specific HPSS Options** section in the HPSS S3 Interface Guide for more details.

2.2.2. SSM API

HPSS now provides a REST API-based interface to SSM. This interface provides a fast, easily programmable, robust interface to monitoring and managing HPSS that facilitates automation.

While this interface does not implement all features of SSM in this release, it supports the most common activities surrounding monitoring and configuration. The remainder of the HPSS functionality will be added in upcoming releases.

Some example endpoints:

- **/latest/servers/status** - Retrieve all server status
- **/latest/health-and-status** - Retrieve HPSS health and status summary
- **/latest/devices-and-drives** - Get status for all devices and drives

For documentation on setup, specific endpoints, and more, see Quick Setup for SSM API.

For an overview of the technologies involved, see Overview.

As a part of this new service, sites will need to identify an OpenID Connect (OIDC) provider. This is a service that provides web-based authentication tokens for use by HPSS web services. Sites may have their own provider, or may use HPSS-provided configuration scripts to install and configure **keycloak**, which is an open source OIDC service. For more details and setup instructions, consult:

- Web Service Prerequisites
- Install and build HPSS
- Configuration of SSM API

Sites should configure SSM API logging using the `$HPSS_ROOT/config/configure_logging` tool.



The HPSS Java-based SSM is planned to be deprecated in an upcoming release, and then removed in the next major release. Sites may begin to see some features which are only available via the SSM API and Web SSM.

2.2.3. Web SSM UI

HPSS now provides a web-based interface to the current SSM status. This provides an easy, modern interface for checking SSM status from any browser.

This web interface currently provides read-only access to a limited set of SSM statistics in this release. Additional views of SSM information and the ability to change SSM configuration changes will be added in upcoming HPSS releases.

Statistics displayed in the current Web SSM UI :

- Server status
- Storage use
- PVL Jobs
- Device and drive usage
- Device and drive status
- Active Storage Classes
- RTM Jobs
- Alarms and warnings

For an overview of the interface, see Using the Web SSM UI.

2.2.4. Support for LTO-10

Support added for LTO-10 tape drives and media.

Update the *Device and Drive List Preferences* window and select *IBM - 3580 Gen10 Tape* to ensure LTO-10 drives are displayed on the *Devices and Drives* window.

2.3. Features retired in 11.3

None.

2.4. Features deprecated in 11.3

None.

2.5. Changes to existing HPSS features in 11.3

2.5.1. HPSS GateKeeper Attributes

The default behavior for sending attributes to the GateKeeper has changed. In the past, the Core Server passed the UID/GID that came from the authorization credentials of the user that was logged in. The new behavior will instead pass the UID/GID for the file we are operating on. In particular, the GID will be passed from the parent directory when the SET_GID bit is set on the parent.

If the site would prefer the old functionality, they can set the HPSS_CORE_GK_USE_LEGACY_ATTRS=1 environment variable in the env.conf file.

2.5.2. Apply Config Logging

The "Apply Config" button in SSM will now issue a log of the changes being applied. This is done using the Config History feature and is the same as running:

```
/opt/hpss/bin/hpss_cfg_history -a -T _LAST APPLY TIME_
```

When the "Apply Config" button is pressed, SSM will log a message like:

```
Nov 21 12:16:13.193167 hpss_ssmsm(HPSS)[1672795]: #msgtype=EVENT #server=SSM System
Manager@hpss #func=ssm_sm_adm_apply_config.c/sm_adm_apply_config/377 #rc=0
#msgid=SSMS0227 #reqid=00000000000000000000000000000000 #msg=SSMS0227 Applying HPSS
configuration changes succeeded, changes: {"since": "2024-11-20-12.15.11", "core":
{"5": {"MAX_ACTIVE_IO_REQS": {"old": 100, "new": 2000}}}, "global": {"PENDING":
{"old": false, "new": true}}}
```

2.6. API changes in HPSS 11.3

2.6.1. Stage Callback Changes

The *hpss_StageCallback*()* API now requires a valid callback address - the request will be rejected if the callback address cannot be resolved by the HPSS Core Server. A new flag (**BFS_NO_CB**) has been added if the caller does not wish for HPSS to actually perform the callback.

A new API was added to allow the caller to obtain the completion status for the asynchronous stage request:

```
int
hpss_GetCallbackStatus(
int              SubsystemId,   /* IN - Subsystem Id */
hpss_reqid_t     CallBackId,    /* IN - Callback Id */
hpss_cb_status_t *Status);      /* OUT - Batch Status */
```

This API has several advantages over the hpss_GetAsyncStatus function - better performance, and less need for bookkeeping on the client. The hpss_GetAsyncStatus function is still useful if there is a need to get status on a small set of files.

Additionally, entries can now be added onto the same callback via multiple calls to the *hpss_StageCallback*()* API. This allows applications to add more stage requests to the same callback over time rather than as a one time activity, and also allows for more total requests to be associated with the same callback ID.

See the **HPSS Programmers Reference** for details.

2.6.2. User Attribute Changes

Several get UDA APIs now return ENOATTR instead of ENOENT when no attributes exist. This clarifies the case where a file exists but has no UDAs.

The affected APIs include:

- hpss_UserAttrXQueryGet
- hpss_UserAttrXQueryGetHandle
- hpss_UserAttrGetAttrs
- hpss_UserAttrGetAttrHandle
- hpss_UserAttrListAttrs
- hpss_UserAttrListAttrHandle

Site tools relying on the old behavior may need updates.

2.6.3. New Python API Bindings

Python bindings have been added for the following APIs:

- hpss_StageCallBackBitfile
- hpss_GetAsyncStatus
- hpss_StageBatch
- hpss_StageBatchCallBack
- hpss_GetCallbackStatus

See *$ python -m pydoc hpss.clapi* for details.

2.6.4. Tape flag for library down

Applications using the HPSS volume level information will see a new flag on the tape volume when the tape is in a PVR which has been marked disabled.

The *pv_list_element_t* Flags structure has a new flag, called *PV_SERVER_DISABLED*. Applications may want to avoid issuing stage or read requests for files in this state as the requests will either be

rejected or paused until the server is enabled again.

2.6.5. HPSS DB diagnostics tool

A new tool has been created, hpss_db_diag(7). This tool leverages the built-in monitor routines for Db2, which provides a comprehensive view of real-time performance metrics without needing to make complex database queries.

This tool provides information in both a human-readable and JSON format.

2.7. New features in 11.2

New features for 11.2 are described below.

2.7.1. Client Striped IOD

HPSS already supports the capability for a client to specify that data transfers utilize multiple TCP/IP connections. However, this has only previously been beneficial when paired with a striped volume.

New in this release, is the ability for HPSS transfers to leverage multiple TCP/IP connections to improve performance for non-striped volume I/O. This can improve transfer performance especially in situations where transfers are bottlenecked by the network transfer time.

This capability is currently supported via:

1. The **PIO API** (see the **HPSS Programmers Reference** for details)
2. The new **Simplified PIO API** (see the Simplified PIO Interface section and **HPSS Programmers Reference** for details).
3. The **hpsscp** and **lori** tools (see man pages for details)
4. The **Parallel FTP (PFTP) Client** (see the **HPSS Users Guide** for details)
5. The **HPSSFS-FUSE Client** (see hpssfs(7) and the **HPSSFS-FUSE Admin Guide** for details)

Additional interfaces will be supported in future releases.

The number of sockets to use for a given client will be site-specific and should be determined by testing and tuning.

2.7.2. LibAIO Support

HPSS now provides the ability to configure Linux asynchronous I/O support (LibAIO) for HPSS disk devices. The Linux AIO capability can often provide improved performance for devices requiring higher levels of parallelism to maximize throughput. See the Configure a new device and drive section for details.

2.7.3. S3 Kerberos Support

The S3 API is specified to use signatures based on shared secrets to authenticate documents.

However, some sites do not allow shared secrets (passwords) to be stored in databases or files. To address this, the HPSS S3 proxy now supports another option: Kerberos. This allows use of one's credential without saving plaintext passwords into a database.

See the Setting up for Kerberos use section for more information on limitations and setup.

2.8. Features retired in 11.2

None.

2.9. Features deprecated in 11.2

None.

2.10. Changes to existing HPSS features in 11.2

The UNIX security framework has been overhauled to use modern cryptography and better support keytabs when used with external authentication mechanisms. This includes the addition of a login service to act as a central security mechanism.

The UNIX GSS mechanism has been rewritten around this login service to allow for more secure authentication. The UNIX mechanism now uses the login service to sign tickets in a way that only the intended user can verify.

With the login service in place, the requirement to set a symlink to the HPSS config data in /etc/hpss or set the home directory of the *hpss* user no longer applies. The UNIX utilities no longer require root permissions for user-level operations, so they are no longer setuid root.

2.11. API changes in HPSS 11.2

2.11.1. Simplified PIO Interface

A new set of APIs has been added to simplify using the PIO transfer interface. PIO is extremely flexible and can handle a wide variety of transfer scenarios, but is not trivial to implement. This simplified set of APIs is meant to cover the common use case of a transfer between HPSS and a local file descriptor using PIO in a way that is as simple as using hpss_Read or hpss_Write.

The new APIs are:

```
int
hpss_PIOReadFile(
   int                      hpss_fd,         /* IN - HPSS file for reading */
   uint64_t                 data_length,     /* IN - Amount of data to read */
   int                      local_fd,        /* IN - Local file for writing */
   uint64_t                *bytes_copied,    /* OUT - Bytes copied */
   hpss_read_queue_item_t  *item);           /* IN - Read Queue Item (optional) */
```

```
int
hpss_PIOWriteFile(
   int                     hpss_fd,         /* IN - HPSS file for writing */
   uint64_t                data_length,     /* IN - Amount of data to write */
   int                     local_fd,        /* IN - Local file for reading */
   uint64_t               *bytes_copied);   /* OUT - Bytes copied */
```

Limited aspects of the PIO transfer such as block size and stripe width can be controlled either via the environment (e.g. **HPSS_API_PIO_STRIPE_WIDTH**) or programmatically.

See the **HPSS Programmers Reference** for more information.

2.11.2. Client Striped IOD in PIO

The PIO *ClntStripeWidth* field in *hpss_pio_params_t* now enables the Client Striped IOD feature of allowing multiple active sockets per HPSS storage device. For example, setting this to 2 will use two sockets instead of one. See the **HPSS Programmers Reference** for more information.

2.12. New features in 11.1

New features for 11.1 are described below.

2.12.1. HPSS Recurse tool

A tool that will traverse an HPSS directory in a multithreaded manner and run a callback with callback-specific parameters on each file and directory in the directory tree. See hpss_recurse(7) for more details.

2.12.2. VV Selection Priority (CR 411)

HPSS can now modify the order in which VVs will be considered for new volume selection. Prior to this CR, the order in which VVs were selected was defined by their creation order. With this change, administrators can now use provided scripts to prioritize use of VVs according to a weighting criteria, or to explicitly customize the order of VV selection.

The primary interface to this feature is a set of scripts, hpss_update_priority_by_pvr(7), hpss_update_priority_by_pvr_drive(7), and hpss_update_priority_by_create_time(7). These scripts take a storage class as their argument, and can also take a configuration file. The configuration file is a JSON file which defines the priority to assign to PVRs and drive types. See the respective man pages for details.

Here is an example configuration file with two PVRs where our criteria is PVR and drive type. This is meant to be used with hpss_update_priority_by_pvr_drive(7). We could do something similar using only PVR weights.

```
{
    "SamePVRPenalty": 2,
    "Weights": [
```

```
        {
            "Name": "PVR7/LTOGEN5",
            "Weight": 5
        },
        {
            "Name": "PVR7/LTOGEN6",
            "Weight": 1
        },
        {
            "Name": "PVR11/LTOGEN5",
            "Weight": 1
        },
    ]
}
```

Once the configuration file has been created, a site will run this update script that applies their desired selection policy periodically to apply the weights to all tape storage classes. Each storage class and/or file family could have a different weighting criteria.

The current priority of VVs can be viewed by using the dump_sspvs(7) tool with the -P option, or by using the new hpss_vv_select_priority(7) script. The hpss_vv_select_priority(7) script can be used to create more complex or custom prioritization schemes or to reset VV priorities.

See the Prioritize Tape Resource Selection and hpss_vv_select_priority(7) sections for details.

2.12.3. Purge By Largest File First

HPSS now provides a capability to purge files sorted by largest file first. When enabled, the purge candidate list is obtained sorted by file size in descending order. All purge candidates are still subject to the purge policy. Note that the file size is not necessarily the number of bytes in the file as files can be written sparsely with gaps.

2.12.4. Secondary Data Copies in AWS

An S3 object store storage class can now be configured as an HPSS storage hierarchy tier, allowing for low-cost secondary data copies to be stored in AWS. The solution is targeted to customers who wish to retain both a local primary data copy and an off-site data copy to address disaster recovery scenarios. Unlike disk and tape storage tiers, data stored at the S3 object store storage class tier can neither be migrated or purged.

2.13. Features retired in 11.1

HPSS features retired in 11.1 are described below.

2.14. Features deprecated in 11.1

HPSS features deprecated in 11.1 are described below.

2.15. Changes to existing HPSS features in 11.1

HPSS features significantly altered in 11.1 are described below.

2.15.1. Updates to RTM Summary List window and rtmu tool

The RTM Summary List window has been updated to show more and finer-grained volume and PVL job information. It now reports multiple read and write volumes and PVL job read and write IDs separately. These appear as new columns in the window. PVName becomes PVNamesRead and PVNamesWrite, and PVLJobId becomes PVLJobIdRead and PVLJobIdWrite. See the field descriptions in the RTM Summary List section. Similar updates have been made to the rtmu(7) command-line tool.

2.16. API changes in HPSS 11.1

All API changes are documented in the *HPSS Programmers Reference*.

Chapter 3. Release 10

This chapter summarizes HPSS changes for Release 10 into four categories: new features, retired features, deprecated features, and changed features. Changes since release 9.3 are described. For changes prior to 10.1, consult prior release notes.

3.1. New features in 10.3

New features for 10.3 are described below.

3.1.1. TS1170 / 3592-70F/S / Jag7 support

Support added for TS1170 / 3592-70F/S / Jag7 media/drive type.

3.1.2. dumpv_pvl can display the PVR name

dumppv_pvl(7) can display the PVR name (or N/A) for each volume by specifying the -r option.

3.1.3. Migration and Purge State Changes

Migrations and purges will no longer always immediately try to start when the Migration/Purge Server (MPS) starts up.

When the MPS starts up, it will check for an environment variable called **HPSS_MPS_MIGR_PAUSE_SECS**. If it exists, it will use that value as the number of seconds for each storage class migration/purge run to wait before checking to see if it needs to run. If the environment variable is not set, it will use a default value of 300 seconds (5 minutes). Valid values for **HPSS_MPS_MIGR_PAUSE_SECS** are 0 to 2147483647. If the administrator wants to change the state of the active storage class migration/purge before the pause is expired, they may do so. For example, they may want to immediately start a migration.

Additionally the MPS will preserve the state of the storage class migration/purge between restarts of the MPS. For example, if *Storage Class 1* migration policy was `Suspended` and then the MPS shutdown, then *Storage Class 1* migration will remain `Suspended` when the MPS starts back up. There is one exception: if a storage class' migration or purge is `Running` when the MPS crashes, then the state will be changed to `Waiting` to wait for the next **Runtime Interval** before starting.

3.1.4. HPSS S3 interface

HPSS now offers an S3 (Simple Storage Service) interface that provides functionality similar to the AWS S3 interface. See the HPSS S3 interface for more information.

3.1.5. Read Queue APIs

A new set of APIs has been added to HPSS to manage the recall of large numbers of files. These APIs are collectively called "Read Queue APIs". See the *HPSS Programmers Reference* for details.

The read queue APIs enable HPSS client applications to send a large list of files to be read to HPSS in a "read queue". The read queue can be added to over time. The client application will request the core server to tell it when a file on its list is ready to run. This enables clients to simply manage very large lists of read requests.

The lori(7) tool implements the read queue APIs described above.

3.1.6. Persisted Read Queues

The read queue APIs are now persisted in HPSS. When a new read queue comes in, it is now persisted in Db2. When entries are added, those entries are persisted in Db2. This allows the core server to recover from downtime and continue processing read queue requests.

The lori(7) tool now supports recovering a read queue in the case that the lori(7) tool crashes using the read queue context ID.

Additionally, there is a new tool to manage persisted read queues. This tool, rq_cleanup(7), will remove either a specific persisted read queue or all read queues. This can be useful in a case where a site goes down for an upgrade and does not want to recover read queue items upon startup.

3.1.7. Updates to avoid server restarts

Updates have been made throughout HPSS to avoid most server restarts after making configuration changes. There is now an *Apply Config* button on the *HPSS Health and Status* window that becomes enabled when configuration changes have been made. After making configuration changes such as adding new migration policies, new purge policies, new storage classes, new hierarchies, and new classes of service, press the *Apply Config* button to apply the changes. Previously, the MPS and Core Servers would need to be restarted to apply these changes, but now this is no longer necessary. See the HPSS Health and Status for more information.

3.1.8. Added purge filters based on file size and age (CR 558)

HPSS now provides a capability to filter purge candidates based on the file size and the age of the file. Files are not subject to purging when two new conditions are met: file size is less than or equal to the specified size in bytes and file last access time is greater than or equal to the given time in minutes. The site administrator will determine the cutoff size and time in the purge policy.

When file size and last access time values are set to zero, the purge exemption is disabled, and the behavior is as it was in previous versions of HPSS.

3.2. New features in 10.2

New features for 10.2 are described below.

3.2.1. Purge on migrate (CR 231)

HPSS now provides a capability when creating files on disk to purge them as part of a successful migration. The new client function hpss_PurgeOnMigrate may be called with a flag of PURGE_ON_MIGRATE_SET to enable this functionality or called with a flag of

PURGE_ON_MIGRATE_CLEAR to disable the functionality. By default, the functionality is disabled. When the PURGE_ON_MIGRATE_SET flag is set, a file that is at level 0 disk that has an associated purge policy will be purged following a successful migration.

3.2.2. Reinit SCSI PVR control paths (CR 609)

The SCSI PVR now supports rebuilding its control path list during reinitialization. This will drop any broken control paths from its list and add any newly available control paths. This will not impact operations for control paths which currently in use.

3.3. New features in 10.1

New features for 10.1 are described below.

3.3.1. HPSS visualization and monitoring (CR 410)

The approach that HPSS has taken to monitoring is to facilitate integration by providing a suite of tools which can be used by administrators to gather information about HPSS into their preferred workflow. A number of additional tools and logging information have been added, which are documented here, to present raw statistical information captured by HPSS.

A site which does not have existing visualization infrastructure for HPSS can use our provided Kibana dashboards to quickly obtain useful HPSS monitoring. However, a site which does have existing visualization infrastructure might benefit from additional sources of data and integrate it into their monitoring.

Beginning in 10.1, HPSS is delivering templates for use with Kibana to provide more trend views. This includes server status, core server statistics, file counts and bytes by class of service or storage class, tape mounts, transfers by device and mover, and trashcan, migration, and purge statistics over time.

See the Installation and configuration of the Elastic (ELK) Stack section for more information.

3.3.2. Restricted access (CR 625)

HPSS now provides finer-grained access controls than the "Restricted User" feature. The "Restricted User" feature was an on/off switch that granted or disabled access to HPSS APIs for particular users. Restricted Access replaces Restricted User as the method for controlling HPSS client access.

Restricted Access allows for a site to restrict access to specific operations by user. Restricted Access duplicates the behavior of the Restricted User feature, but also provides finer-grained control by operation such as restricting copies, creates, writes, and stages.

See Restricting user access to HPSS for more information.

3.3.3. Files now show the type of media in extended attributes

Extended attribute calls (e.g. hpss_FileGetXAttributes) which retrieve level information now include the type of the media, for each tape reported. The media type can be converted to a more useful

string with hpss_MediaTypeString(). See the *HPSS Programmers Reference* for more details.

3.3.4. Files now show current activity in extended attributes

Extended attribute calls (e.g. hpss_FileGetXAttributes) now report back current activity against the file. This includes reads, writes, migrates, purges, changecos, and stages. A value of -1 indicates that the file is not open. See the *HPSS Programmers Reference* for more details.

3.3.5. Low overhead read interface (lori - CR 562)

lori(7) is a new tool which behaves similarly to quaid and shares many of the same options including organizing requests by tape, callouts upon read completion, filtering, and more. However, while quaid stages files up to the top level of the hierarchy, lori uses PIO to read data out of HPSS with minimal overhead and into the selected output directory.

In the future, lori will take advantage of upcoming changes to enable mass recall with read recovery.

3.3.6. dump_acct_sum will now dump the bandwidth table

dump_acct_sum(7) will now present the bandwidth table (bytes read and written) by account/UID/GID/COS using the -b option.

3.3.7. dumpv_pvl now displays the HPSS label format

dumppv_pvl(7) now outputs the tape label format for each volume.

3.3.8. lscos and lsvol now support JSON output

lscos(7) and lsvol(7) now support outputting in JSON format with -j.

3.3.9. Toggle RAO on/off per tape device

RAO can now be toggled on/off for each tape device. The default is ON. See the Configure a new device and drive for more information.

3.3.10. Repack will log media access INFO logs

repack(7) now logs information about source and destination tapes involved in repack operations as INFO logs. Individual file access is not logged for repack.

3.3.11. HPSS server metrics tool

A new tool has been created, hpss_server_metrics(7). This tool provides system statistics which previously could only be gathered through a variety of other tools and APIs.

The server metrics tool provides statistics in both human-readable and JSON format, and can also reset statistics for certain reports to generate interval-based statistics.

3.3.12. HPSS DB metrics tool

A new tool has been created, hpss_db_metrics(7). This tool provides database statistics which previously could only be gathered through out of band database queries.

The db metrics tool provides statistics in both human-readable and JSON format.

3.4. Features retired in 10.1

HPSS features retired in 10.1 are described below.

3.4.1. Restricted user list

The Restricted Access feature replaces the Restricted User List feature. The Restricted User screens in the SSM have been removed.

The Restricted Access feature can replicate the behavior of the Restricted User List by denying access to the same set of users for "ALL" operations.

See Restricting user access to HPSS for more information.

3.5. Features deprecated in 10.1

There were no HPSS features deprecated in 10.1.

3.6. Changes to existing HPSS features in 10.1

HPSS features significantly altered in 10.1 are described below.

3.6.1. HPSS now supports 65535 alternate groups

HPSS now supports users belonging to 65535 alternate groups, up from 64.

3.6.2. hpssmsg now supports more fields

hpssmsg(7) is a tool to log messages in the HPSS log message format. Previously, this tool could only log a message with a type and severity. Now, hpssmsg also allows return code, functions, request id, program name, and a message ID to be logged. See the hpssmsg man page for more information.

3.6.3. Migration/Purge server raw report files now using JSON Lines.

The Migration/Purge Server writes daily reports. The reports have been changed to newline JSON formatted records. The new format is using JSON Lines (https://jsonlines.org/) for each record. This new format allows other tools like ELK and Splunk to take advantage of JSON output.

The mps_reporter(7) tool has been updated to work with the new format. However, to view older MPS reports, you must use the mps_reporter_legacy(7) tool.

When upgrading to HPSS 10.1, archive all files in the MPS report path before starting the 10.1

Migration/Purge Server. The old format of the MPS Report files are not compatible with the new JSON Lines format.

3.6.4. JSON timestamps now in ISO 8601 format

Timestamps in HPSS output to JSON widely used "seconds since epoch" format. However, for ease of use with applications that process timestamp data, the standard ISO 8601 format has been selected.

For example:

Epoch timestamp: 1656091032

Equivalent ISO 8601 formatted date: 2022-06-24T12:17:12Z

This change in the JSON timestamp format does not impact non-JSON outputs for these tools.

The following tools were modified to use the ISO 8601 format:

- dump_acct_sum(7)
- lsvol(7)
- lshpss(7)
- rtmu(7)
- dump_sspvs(7)
- lspvhist(7)

3.6.5. Parallel quaid file scans

quaid(7) now performs its initial input selection scans in parallel. This allows quaid to organize the inputs and begin staging faster. Users can control the amount of parallelism of the scan using the -n option.

3.6.6. Avoid purging newly staged data with Last Access Time purge policies

When purge records were created following a stage request they used the file access statistics to create the purge time. When combined with the last access time purge policy, this would mean that a staged file was immediately purgeable since the file access time had not been updated.

HPSS has been modified to use the internal statistics for the top level when creating the purge record. This timestamp will be the time that the data was staged, meaning that the data will not be eligible to be purged until it has reached the Last Access Time threshold.

3.7. API changes in HPSS 10.1

All API changes are documented in the *HPSS Programmers Reference*.

- The functions hpss_GetRestrictedUserList and hpss_ReadRestrictedUserFile were removed.
- A new function, hpss_GetUID, was added to get the current client thread's UID.

- A new function, hpss_GetGID, was added to get the current client thread's GID.
- A new function, hpss_SetApplicationName, allows an application to set its name. This name will be logged at the core server REQUEST logs.
- The sec_cred_t structure has been modified to include a dynamically allocated array of that supports 64K alternate groups. Functions which return sec_cred_t now return heap allocations that need to be freed. Uses of sec_cred_t in HPSS to deal with credentials should be handled with care when porting applications to HPSS 10.1 to deal with memory leaks. The functions hpss_SECFreeCreds and hpss_SECDuplicateCred have been added to facilitate these changes.
- A number of APIs have been added for "read queueing". These APIs are not currently implemented and return ENOTSUP.

Chapter 4. Release 9

4.1. New features in 9.3

New features for 9.3 are described below.

4.1.1. HPSS abort I/O tool (CR 546)

A new tool hpss_abort_io(7) is now available. The hpss_abort_io(7) client command-line tool facilitates aborting I/O requests in HPSS. The tool has the following features:

- Abort a list of requests from the command line, or via stdin
- Abort requests ID using a **partial** request ID string (four or more characters)
- Abort all requests for the authenticated user
- For empowered users, abort all requests for a specified user
- For empowered users, abort all requests associated with a Mover

4.1.2. Calculate the atime

The Core Server sets the atime in directories and symlinks. There are three different ways that the atime can be calculated. These three different algorithms are described in the Management Guide. Administrators can choose one of these algorithms (NoATime, StrictATime, or RelativeATime) by specifying their choice in environment variable HPSS_CORE_ATIME_POLICY. If no atime algorithm is specified in environment variable HPSS_CORE_ATIME_POLICY the default algoritm, RelativeATime, is used to calculate the atime.

4.1.3. Support for Mover for fast positioning

The Mover has a new positioning mode, which enables it to skip verifying tape section headers and position directly to requested data. Read operations at the beginning of the section will still verify the tape section header.

Verifying the section header provides additional assurance to HPSS that the Mover has correctly positioned; however, modern tape drives have an excellent track record of correctly positioning the media. When reading multiple files within aggregates, or issuing small file reads within a large file on tape, header verification during positioning can take significant time and generate additional wear on media.

By default, this mode is OFF and positioning continues to work as it has previously.

To begin using the fast positioning mode, set the "Fast Positioning" flag in the tape device to ON via SSM. See Device and drive management for details.

Another setting that modifies this behavior is HPSS_MVR_LOCATE_BLK_THRESH (default = 0). This setting causes the Mover to go ahead and verify the tape header position so long as the requested position is within some number of blocks of the tape header. For example, if HPSS_MVR_LOCATE_BLK_THRESH is set to 100, the header will be verified if the block being

positioned to is within 100 blocks of the header. This is useful, for example, to allow the header position to occur in a scenario where the tape header was very close to the requested data in order to have a finer-grained caution / performance trade-off, if desired. After altering the environment variable, the Mover must be restarted for any changes to take effect. The effect of this is node-wide.

4.1.4. Parallel transfer abort in PFTP

The HPSS PFTP server has added features that allow users to abort parallel file transfers in the PFTP client. The following commands have been added to the PFTP server:

SITE ABORTREQ [request ID]

The PFTP server will abort the file transfer corresponding to the provided request ID.

SITE GENREQID

The PFTP server will return a request ID that will be used with the next parallel (**pput**/**pget**/PSTO/PRTR) file transfer initiated by the user.

SITE REPORTREQID

This command is a toggle that is defaulted to "off". When toggled "on", the request ID associated with a given transfer is reported during the transfer.

4.1.5. Defer tape labeling on import (CR 607)

The ability to label a tape when it is first needed for a write, instead of when it is first imported into the HPSS system, is now supported. The labeling behavior, specifically, for these deferred label tapes follows the same behavior as tapes that are labeled immediately, and across all import types (DEFAULT, SCRATCH, and OVERWRITE).

By default, this feature is disabled, meaning tapes will be labeled immediately at the time of import. To enable the feature (for a particular import session), the administrator can select "Defer Labeling" from the "Import Tape Volumes" configuration screen.

To view the label format of the physical volume, the administrator can run the "lshpss -vol" command. Deferred label tapes will specify one of: Defer (D), Defer (S), or Defer (O) to signify one of Deferred Label Default Import, Deferred Label Scratch Import, or Deferred Label Overwrite Import, respectively.

See Import Tape Volumes window for more information.

4.1.6. Updates to hpss_log_archive

The hpss_log_archive(7) tool has been updated to use a much larger default block size of 100 MB vs 1 MB before. Also, users may now override the default block size with the use of the -b argument or by setting the HPSS_LOG_BLKSIZE environment value. Users may specify a numerical size along with a unit size specification (K/KB, M/MB, G/GB, etc.) for convenience. For example, if -b 1G were specified, then the block size used would be 1 GB.

4.1.7. Disk Mover & VV I/O Metrics

The HPSS Core Server's Storage Service maintains metrics about how busy disk movers and disk VVs are. These metrics measure how many I/O tasks (reads and writes) are currently running against the disk movers and their disk VVs. These metrics are used to load-balance disk allocations across the available movers and VVs.

Prior to HPSS 9.3, these disk mover and VV I/O metrics were entirely internal to the Core Server. As of HPSS 9.3, the metrics can be viewed externally in the Core Server state dump and via the `showdiskmaps` tool. This can be especially useful during debugging sessions or when investigating disk load-balancing problems.

The new functionality consists of two related features:

- A log of all the allocations, reads, and writes and how they impact the metrics. This info is available in the Core Server state dump — obtained via sending `SIGHUP` to the `hpss_core` process.
- A point-in-time snapshot of disk and mover I/O metrics. This is available in the `showdiskmaps` tool's output. **NOTE**: the version of `showdiskmaps` provided with HPSS 9.3 will **not** be able to read disk free space map snapshot files from prior versions of HPSS.

See the relevant disk VV I/O metrics section see HPSS Disk Mover and VV Load-Balancing for more details.

4.1.8. New location for HPSS man pages

The HPSS man pages were previously installed under a system directory. Now they are located under a directory where the rest of the HPSS files reside. The new location supports having RPMs for different HPSS versions installed at the same time. In order to properly view the man pages, some adjustments may need to be made to system settings. Refer to Manual pages setup for specific information.

4.2. Features retired in 9.3

HPSS features retired in 9.3 are described below.

None.

4.3. Features deprecated in 9.3

HPSS features deprecated in 9.3 are described below.

HPSS GAS-API. HPSS now uses the GSSAPI directly. Any applications which used the GSSAPI should switch to the standard GSS calls instead.

4.4. Changes to existing HPSS features in 9.3

HPSS features significantly altered in 9.3 are described below.

4.4.1. Increased COS and storage class limits

The COS and storage class limits have been increased from 64/192 to 1024/3072. There are no related limits to hierarchies, migration policies, purge policies, or other similar constructs.

4.4.2. JSON output for dump_sspvs

The dump_sspvs(7) tool now supports JSON output. Use the **--json (-j)** option to request JSON-formatted output.

4.4.3. Users can now pass a VV name to dump_sspvs

The dump_sspvs(7) tool now allows users to provide a virtual volume name to process via a new **--vv-name (-v)** option. The tool will allow wild card values "*" and "?" for multiple or single character substitution respectively. Only VV names that match the pattern or exact name will be reported when this new parameter is used.

4.4.4. Pre-queued stages for mass stage

During stage requests from tape to disk, HPSS now avoids firing off the disk Mover request until the tape drive is ready to process the read. This addresses several issues that may have previously occurred during mass stage events (for example, staging hundreds or thousands of files) such as:

1. Disk Mover stage timeouts during mass stage operations, caused because the disk Mover was idling while their partner requests for tape access were waiting in the tape queue
2. Running out of disk Mover connections due to mass stage operations, caused because disk Movers were idling as above
3. Disk contention issues upon mass stage operations, caused because the disk I/O was scheduled at the time the disk mover was contacted, which could be hours before the tape I/O was able to run

4.4.5. PFTP now supports direct I/O

PFTP now supports reading local file data with O_DIRECT. This is useful for bypassing kernel buffering. This can be enabled by adding "Use DirectIO" to the PFTP Client section of `HPSS.conf`. Note that O_DIRECT will only be used if the transfer is block-aligned. The local system block size can be set by adding "DirectIO Block Size = <Filesystem Block Size>" to the PFTP Client section of `HPSS.conf`.

4.4.6. Detect and avoid drives being cleaned

HPSS will now detect and avoid tape drives which indicate they are in imminent need of cleaning. HPSS will query the drive upon unload to determine whether cleaning is necessary. If so, the PVL will begin requesting that the PVR avoid that drive on future mounts if there are other drives available to service the request. Additionally, the PVL will log a debug-level log when it detects that the drive requires cleaning or that it no longer requires cleaning.

This enhancement allows for cleaning to occur for as long as possible until the drive is needed, rather than being interrupted when other drives are available.

4.5. New features in 9.2

New features for 9.2 are described below.

4.5.1. Support for Spectra Logic TAOS

HPSS now supports the Spectra Logic feature "TAOS" (Time-Based Access Order System). TAOS provides an interface that functions similarly to RAO on TS11xx hardware, but for LTO-7 and LTO-8 tape drives. The TAOS interface allows for device-ordered requests via the Spectra Logic tape library to improve performance and reduce wear.

Within HPSS TAOS-enabled drives are treated similarly to RAO devices for recall and stage scheduling. Tape devices may be configured to be TAOS-enabled via SSM. Once the drives are configured to be TAOS-enabled, TAOS will be used by HPSS transparently to the end user similarly to RAO.

See the Configure a new device and drive for more information on TAOS support for tape devices in HPSS.

4.5.2. Support for LTO-9

Support added for LTO-9 tape drives and media.

4.5.3. Increase tape aggregation limits

HPSS now defaults to writing tape aggregates for all disk-to-tape migrations. If for some reason any file cannot be migrated to a tape aggregate, HPSS may instead migrate it as a single file, or HPSS may skip that file for the current migration. See Debugging tape aggregation for more information.

HPSS administrators are no longer limited to a maximum of 1,000 files in an aggregate. The maximum number of files in an aggregate has been increased to 50,000 files. The larger number of files allowed per aggregate means that a larger number of files can be open concurrently during migration, which means that the Core Server configuration may need to be adjusted. See the Core Server I/O limit configuration section for information on how to ensure that the Core Server configuration is set correctly to accommodate the appropriate Core Server I/O limits. See the Core Server max open files configuration section for information on how to ensure that the system is configured for a sufficient number of concurrently open files.

The Disk Migration Policy screen has changed to remove the "Aggregate Files to Tape" flag, "Max Files in Aggregate" field, and "Max File Size in Aggregate" field. A new "Max Aggregate Size" field has been added to limit the total size in bytes of an aggregate. After upgrading from HPSS 9.1 or earlier, review all disk migration policies and storage class configurations to ensure the desired operation.

For very large aggregate sizes, HPSS Disk Movers may begin timing out during the migration. The Mover environment variable MVR_CLIENT_TIMEOUT may need to be increased on the Movers. See *Additional Mover configuration* in the *HPSS Management Guide* for more detail. Hitting the MVR_CLIENT_TIMEOUT is more probable now since HPSS now allows larger aggregates than previous HPSS versions.

The size of an aggregate is configurable by setting the "Max Aggregate Size" field in the disk migration policy. This value is an absolute value, and care must be taken to ensure a correct value if the migration policy is shared between storage classes.

Now that tape aggregates are no longer limited by a maximum number of member files, the repack utility's **--max-files-per-aggregate** / **-F** command-line option has been removed. The only control on aggregate size that remains is **--max-bytes-per-aggregate** / **-L**. To turn off tape aggregation during repack, the user can use **--max-bytes-per-aggregate=0** on the command line. Further, **--max-bytes -per-aggregate** is no longer limited to 10% of the destination VV Estimated Size. The limit is now the full Estimated Size of the destination VV, as configured in the storage class. This means that Repack can generate far larger aggregates than in prior versions of HPSS.

4.6. Features retired in 9.2

HPSS features retired in 9.2 are described below.

None.

4.7. Features deprecated in 9.2

HPSS features deprecated in 9.2 are described below.

None.

4.8. Changes to existing HPSS features in 9.2

HPSS features significantly altered in 9.2 are described below.

The HPSS logrotate template has been modified. Your site may want to review and apply the logrotate changes to your site logrotate policy, or replace the logrotate policy entirely if your site has no local logrotate modifications for HPSS.

The logrotate template has been modified to incorporate the logging of each rotation into hpss_log_archive itself using the HPSS log format. This is done to avoid the need for calls out to hpssmsg(7) in the prerotate and postrotate portions of the logrotate configuration.



In order to properly handle log archive failures, the call to hpss_log_archive must be the final call in each script, or must be wrapped in a script which retains and returns any errors from hpss_log_archive(7) - the exit code of the final script directive determines logrotate's behavior for prerotate and postrotate, any intermediate errors are ignored.

4.9. New features in 9.1

New features for 9.1 are described below.

4.9.1. Provide API tools

HPSS now provides a set of basic API tools for use in HPSS. These tools are in the unsupported tools directory; they are a modified version of the support-provided "Sample Code". See the individual tool man pages for details. The purpose of these tools is to provide a convenient command-line tool suite into HPSS for administrators for ease of scripting and basic system use. These tools fill a niche in HPSS to provide a basic set of tools for interacting with HPSS out of the box.

These tools are not meant to be a high-performance client interface or alternative to such interfaces. There are significant startup costs associated with these tools including authentication and connection management that will keep these tools from performing as well as other applications (such as FUSE, HSI, or PFTP) at scale.

The following tools are provided:

1. hpsscat(7) - Read an HPSS file
2. hpsschmod(7) - Change an HPSS file's mode
3. hpsschown(7) - Change an HPSS file's owner
4. hpsscp(7) - Copy a file into or out of HPSS
5. hpssgetvol(7) - Get volume information about an HPSS file
6. hpssgetxattrs(7) - Get extended attribute information about an HPSS file
7. hpssln(7) - Create a link or symlink in HPSS
8. hpssls(7) - List an HPSS directory
9. hpssmkdir(7) - Make an HPSS directory
10. hpsspurge(7) - Purge an HPSS file
11. hpssrm(7) - Remove an HPSS file
12. hpssrmdir(7) - Remove an empty HPSS directory
13. hpsssetcos(7) - Set the COS of an existing HPSS file
14. hpssstat(7) - Stat an HPSS file
15. hpsstouch(7) - Touch an HPSS file

4.9.2. Automatic Change Class of Service

With this feature, HPSS administrators will be able to enable an Automatic Change Class of Service (COS) operation on a per-COS basis. When this new COS flag is turned on, upon disk migration, files within this COS that fall outside the **Minimum File Size** and **Maximum File Size** will have their COS changed to a target COS that has the appropriate file size range. Some use cases:

- Site has files that come into their default COS without a file size (for example, users running vanilla FTP on their desktop).
- Site is using a COS for an HPSSFS FUSE mount point, **Enforce Maximum File Size** is off, and the file sizes are not within a consistent range.

The Migration/Purge Server will initiate the change in COS which will be performed by the Core Server as a background task. COS's with the **Force Selection** flag on will not be considered for the target COS. If multiple COS's meet the criteria, then the Migration/Purge Server will select the first COS that meets the criteria; if no COS meets the criteria, then the file will remain in the current COS and be migrated according to the current COS's hierarchy. If the Core Server's Specific Configuration **COS Copy to disk** flag is not on and there is a tape level in the target COS's hierarchy, then the Core Server will not automatically copy the file to the disk level and will instead copy to the tape level. To take advantage of migration policies for the file's new COS's hierarchy (like tape aggregation), it is recommended that you turn **COS Copy to disk** on.

4.9.3. Client I/O abort

The HPSS Client API now provides several interfaces allowing client applications to abort user I/O requests. Applications using HPSS can begin using these interfaces to cancel ongoing I/O requests.

hpss_AbortIOByRequestId

HPSS will abort a specific I/O request by request ID. Applications can manage their request IDs and cancel specific requests. Normal users may only abort their own I/O requests, while empowered users may abort any I/O request. See the *HPSS Programmer's Reference* for more information.

hpss_AbortIOByUserId

HPSS will abort **all** I/O requests initiated by the provided user ID, regardless of interface. If no user is provided, the authenticated user credentials will be used. Specifying an alternate user is only allowed by empowered users. This interface reports the number of requests aborted by this call, the number of requests encountered which were previously aborted, and the total number of requests found. Note that requests that have been aborted, but have not completed their abort processing will still be counted as aborted. See the *HPSS Programmer's Reference* for more information.

4.9.4. Account management in PFTP

The HPSS PFTP server and client has added features that allow users to manage accounts inside the PFTP client. The following commands have been added to the PFTP server:

ACCT <account id>

The PFTP server will assign all the activity in this session to the account ID provided.

SITE GETACCT [path]

If a path is provided, the PFTP server will return the account ID assigned to the file at the given path. If no path is provided, the PFTP server will return the account ID associated with the current session.

SITE SETACCT <path> <account id>

The PFTP server will assign the file at the path given to the given account ID.

The PFTP client provides an **account** command that sends the provided account to the **ACCT** command to the server. Like all **site** commands, **site getacct** and **site setacct** input are sent directly

from the PFTP client to the server.

4.9.5. Buffered Tape Marks

The Tape Mover can now be configured to write buffered tape marks. Using buffered tape marks increases the rate at which data will be written to tape by not synchronizing the drive buffer with the medium after every tape mark write. The drawback is that on write errors, the tape will be rolled back to the last known good/synced tape mark, causing the Tape Mover to potentially have to discard and rewrite some amount of good data.

Buffered tape marks are disabled by default and can be enabled via the "Seconds Between Unbuffered Tape Marks" (SBUTM) and "Seconds Between Tape Marks" (SBTM) fields in the Tape Storage Class Configuration menu. SBUTM must be a multiple of SBTM. If SBTM and SBUTM are equivalent, buffered tape marks are disabled.

An example of using buffered tape marks is as follows: an SBUTM value of 180 (seconds) and an SBTM value of 45 (seconds) indicates that a buffered tape mark is written every 45 seconds, and an unbuffered/synced tape mark is written every 180 seconds. In other words, every fourth tape mark will be a synced tape mark.

4.10. Features retired in 9.1

HPSS features retired in 9.1 are described below.

None.

4.11. Features deprecated in 9.1

HPSS features deprecated in 9.1 are described below.

None.

4.12. Changes to existing HPSS features in 9.1

HPSS features significantly altered in 9.1 are described below.

4.12.1. The scsi.diag can now be squelched system-wide

The SCSI diagnostic log file `scsi.diag` may now be forcibly closed by all HPSS processes by creating a `/var/hpss/tmp/noscsidiag` file. HPSS processes logging to `scsi.diag` will stop logging within 30 seconds and close the file. This allows the `scsi.diag` file to be cleaned up or rotated easily. Removing the `/var/hpss/tmp/noscsidiag` file resumes all logging for any HPSS process that is logging SCSI diagnostics.

The lscos(7) tool now supports two summary methods. The basic summary method, **-S**, lists each stream and a count of the bitfiles associated with it. The detailed summary method, **-T**, lists the count for each combination of stream, file family, and destination COS. For example:

```
# lscos -i 1 -S
  Stream ID            Count
          0                3

# lscos -i 1 -T
 Stream ID  Family ID New COS ID            Count
         0          0          2                3
         0          1          2                1
```

Chapter 5. HPSS basics

5.1. Introduction

The High Performance Storage System (HPSS) provides hierarchical storage management and services for very large storage environments. HPSS may be of interest to organizations having present and future scalability requirements that are very demanding in terms of total storage capacity, file sizes, data rates, number of objects stored, and numbers of users. HPSS is part of an open, distributed environment based on remote procedure calls, Kerberos, LDAP directory systems and DB2 which form the infrastructure of HPSS. HPSS is the result of a collaborative effort by leading US Government supercomputer laboratories and industry to address real and urgent high-end storage requirements. HPSS is offered commercially by IBM.

HPSS provides scalable parallel storage systems for highly parallel computers as well as traditional supercomputers and workstation clusters. Concentrating on meeting the high end of storage system and data management requirements, HPSS is scalable and designed to store up to petabytes (10^{15}) of data and to use network-connected storage devices to transfer data at rates up to multiple gigabytes (10^{9}) per second.

HPSS provides a large degree of control for the customer to manage the hierarchical storage system. Using configuration information defined by the customer, HPSS organizes storage devices into multiple storage hierarchies. Based on policy information defined by the customer and actual usage information, data are then moved to the appropriate storage hierarchy and to appropriate levels in the storage hierarchy.

5.2. HPSS capabilities

A primary goal of HPSS is to move large files between storage devices and parallel or clustered computers at speeds many times faster than today's commercial storage system software products and to do this in a way that is more reliable and manageable than is possible with current systems. In order to accomplish this goal, HPSS is designed and implemented based on the concepts described in the following subsections.

5.2.1. Network-centered architecture

The focus of HPSS is the network, not a single processor as in conventional storage systems. HPSS provides servers that can be distributed across a high-performance network to provide scalability and parallelism. The basis for this architecture is the *IEEE Mass Storage System Reference Model, Version 5*.

5.2.2. High data transfer rate

HPSS achieves high data transfer rates by eliminating overhead normally associated with data transfer operations. In general, HPSS servers establish and control transfer sessions but are not involved in the actual transfer of data.

5.2.3. Parallel operation

The HPSS Client Application Program Interface (Client API) supports parallel or sequential access to storage devices by clients executing parallel or sequential applications. HPSS also provides a Parallel File Transfer Protocol (PFTP). HPSS can even manage data transfers in a situation where the number of data sources differs from the number of destination sources. Parallel data transfer is vital in situations that demand fast access to very large files.

5.2.4. Based on standard components

HPSS runs on UNIX and is written in ANSI C and Java. It uses remote procedure calls, a selectable security service (Kerberos or UNIX), UNIX or LDAP for user configuration information, and DB2 as the basis for its portable, distributed, transaction-based architecture. These components are offered on many vendors' platforms.

The full HPSS system has been implemented on the Linux platform, and some components of HPSS are available on other platforms. Refer to HPSS hardware platforms and Prerequisite software considerations. for specific information.

To aid vendors and users in porting HPSS to new platforms, HPSS source code is available.

5.2.5. Data integrity through transaction management

Transactional metadata management, provided by DB2, enables a reliable design that protects user data both from unauthorized use and from corruption or loss. A transaction is an atomic grouping of metadata management functions managed so that either all of them take place together or none of them takes place. Journaling makes it possible to back out any partially complete transactions if a failure occurs. Transaction technology is common in relational data management systems but not in storage systems. It is the key to maintaining reliability and security while scaling upward into a large, distributed storage environment.

5.2.6. Multiple hierarchies and Classes of Services

Most other storage management systems support simple storage hierarchies consisting of one kind of disk and one kind of tape. HPSS provides multiple configurable hierarchies, which are particularly useful when inserting new storage technologies over time. As new disks or tapes are added, new classes of service can be set up. HPSS files reside in a particular class of service which users select based on parameters such as file size and performance. A class of service is implemented by a storage hierarchy which in turn consists of multiple storage classes, as shown in Class of Service/hierarchy/storage class.

Storage classes are used to logically group storage media to provide storage for HPSS files. A hierarchy may be as simple as a single tape, or it may consist of two or more levels of disk and local tape. The user can even set up classes of service so that data from an older type of tape is subsequently migrated to a new type of tape. Such a procedure allows migration to new media over time without having to copy all the old media at once.

5.2.7. Storage subsystems

Storage subsystems (or simply, "subsystems") may be used to increase the scalability of HPSS in handling concurrent requests or to meet local political needs. Each subsystem contains a single Core Server. If migration and purge are needed for the subsystem, then it will also contain a Migration/Purge Server. All other servers are subsystem-independent.

Data stored within HPSS is assigned to different subsystems based on pathname resolution. A pathname consisting of a forward slash ("/") resolves to the root Core Server which is specified in the global configuration file. However, if the pathname contains junction components, it may resolve to a Core Server in a different subsystem. For example, the pathname `/JunctionToSubsys2/mydir` could lead to a fileset managed by the Core Server in subsystem 2. Sites which do not wish to partition their HPSS through the use of subsystems will run HPSS with a single subsystem.

5.3. HPSS components

The components of HPSS include files, filesets, junctions, virtual volumes, physical volumes, storage segments, metadata, servers, infrastructure, user interfaces, a management interface, and policies. Metadata is control information about the data stored under HPSS, such as location, access times, permissions, and storage policies. Most HPSS metadata is stored in tables in a DB2 relational database. Media and file metadata are represented by data structures that describe the attributes and characteristics of storage system components such as files, filesets, junctions, storage segments, and volumes. Servers are the processes that control the logic of the system and control movement of the data. The HPSS infrastructure provides the services that are used by all the servers for operations such as sending messages and providing reliable transaction management. User interfaces provide several different views of HPSS to applications with different needs. The management interface provides a way to administer and control the storage system and implement site policy.

These HPSS components are discussed below in the following sections.

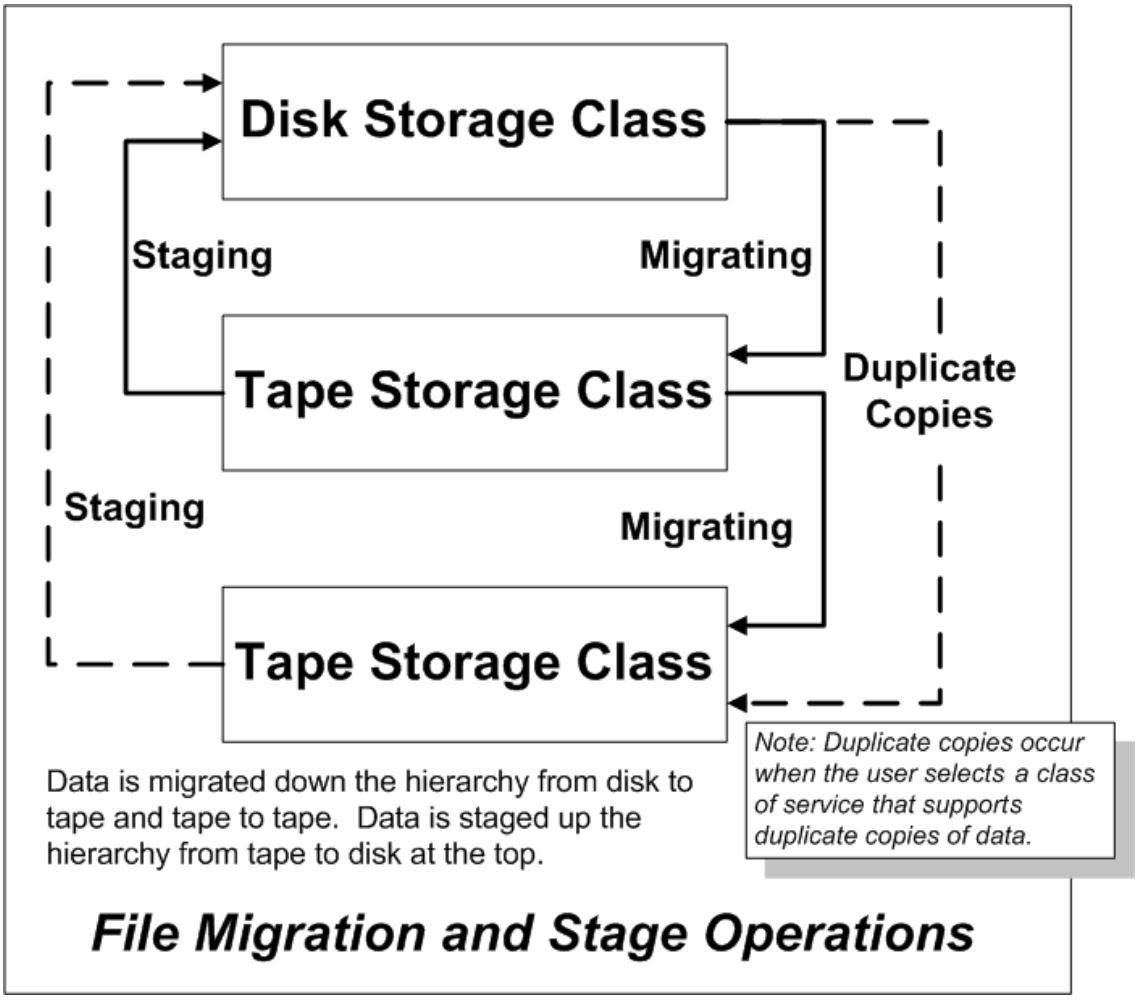


Figure 1. File migration and stage operations

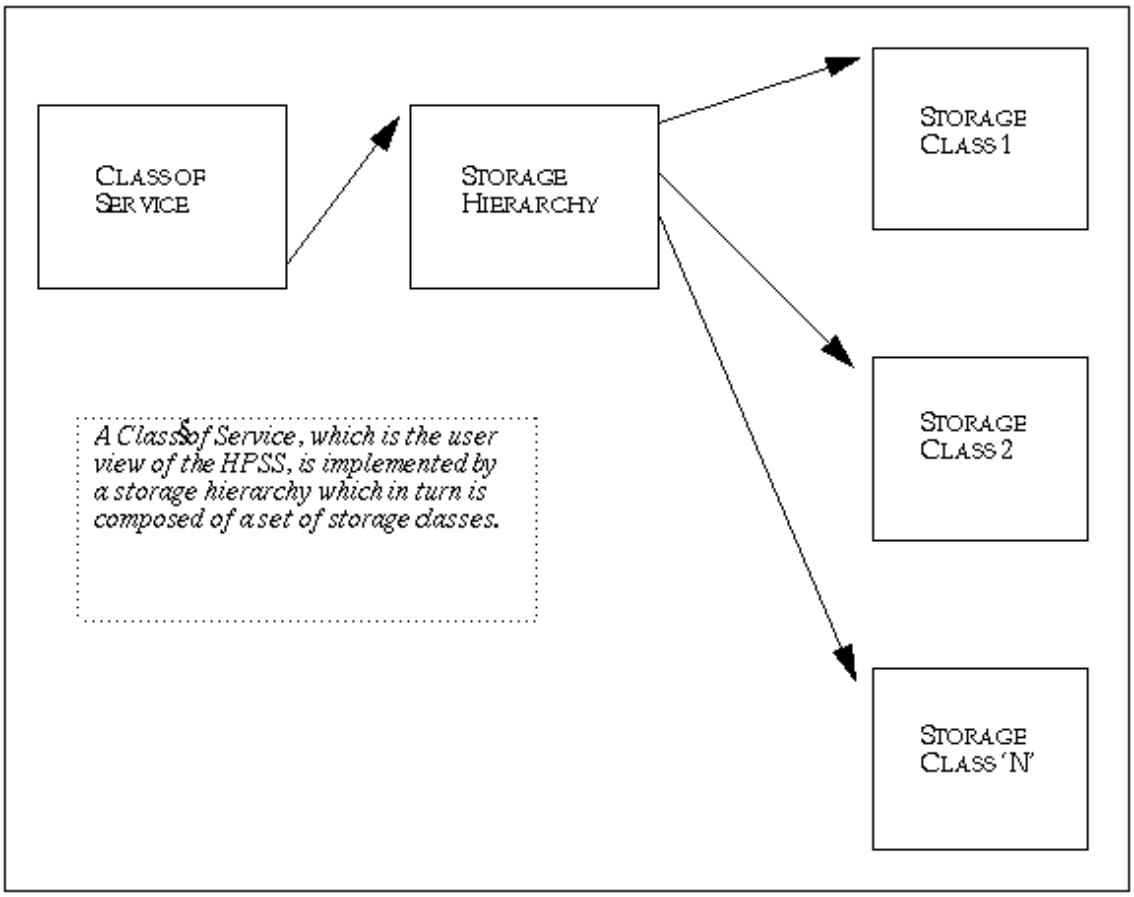


Figure 2. Class of Service/hierarchy/storage class

5.3.1. HPSS files, filesets, volumes, storage segments and related metadata

The various metadata constructs used to describe the HPSS name space and HPSS storage are described below:

Files (bitfiles)

Files in HPSS, called bitfiles in deference to IEEE Mass Storage Reference Model terminology, are logical strings of bytes, even though a particular bitfile may have a structure imposed by its owner. This unstructured view decouples HPSS from any particular file management system that host clients of HPSS might use. HPSS bitfile size is limited to 2^{64} - 1 bytes.

Each bitfile is identified by a machine-generated name called a bitfile ID. A bitfile may also have a human-readable name. It is the job of the HPSS Core Server (discussed in HPSS servers) to map a human-readable name to a bitfile's ID.

Filesets

A fileset is a logical collection of files that can be managed as a single administrative unit, or more simply, a disjoint directory tree. A fileset has two identifiers: a human-readable name and a numeric fileset ID. Both identifiers are unique to a given realm.

Junctions

A junction is a Core Server object, much like a symbolic link to a directory, that is used to point to a fileset. This fileset may belong to the same Core Server or to a different Core Server. When pointing to a different Core Server, junctions allow HPSS users to traverse to different subsystems.

File Families

HPSS files can be grouped into families. All files in a given family are recorded on a set of tapes assigned to the family. Only files from the given family are recorded on these tapes. HPSS supports grouping files on tape volumes only. A family can be selected at the time a file is created by supplying the appropriate family ID as one of the create parameters. All files created in the fileset belong to the family. When one of these files is migrated from disk to tape, it is recorded on a tape with other files in the same family. If no tape virtual volume is associated with the family, a blank tape is reassigned from the default family. The family affiliation is preserved when tapes are repacked.

Physical Volumes

A physical volume is a unit of storage media on which HPSS stores data. The media can be removable (for example, cartridge tape or optical disk) or non-removable (magnetic disk). Physical volumes may also be composite media, such as RAID disks, but must be represented by the host OS as a single device.

Physical volumes are not visible to the end user. The end user stores bitfiles into a logically unlimited storage space. HPSS, however, must implement this storage on a variety of types and quantities of physical volumes.

For a list of the tape physical volume types supported by HPSS, see Tape devices.

Virtual Volumes

A virtual volume is used by the Core Server to provide a logical abstraction or mapping of physical volumes. A virtual volume may include one or more physical volumes. Striping of storage media is accomplished by the Core Servers by collecting more than one physical volume into a single virtual volume. A virtual volume is primarily used inside of HPSS, thus hidden from the user, but its existence benefits the user by making the user's data independent of device characteristics. Virtual volumes are organized as strings of bytes up to 2^{64} - 1 bytes in length that can be addressed by an offset into the virtual volume.

Storage Segments

A storage segment is an abstract storage object which is mapped onto a virtual volume. Each storage segment is associated with a storage class (defined below) and has a certain measure of location transparency. The Core Server (discussed in HPSS servers) uses both disk and tape storage segments as its primary method of obtaining and accessing HPSS storage resources. Mappings of storage segments onto virtual volumes are maintained by the HPSS Core Servers.

Storage Maps

A storage map is a data structure used by the Core Server to manage the allocation of storage space on virtual volumes.

Storage Classes

A storage class defines a set of characteristics and usage parameters to be associated with a particular grouping of HPSS virtual volumes. Each virtual volume and its associated physical volumes belong to a single storage class in HPSS. Storage classes in turn are grouped to form storage hierarchies (see below). An HPSS storage class is used to logically group storage media to provide storage for HPSS files with specific intended usage, similar size and usage characteristics.

Storage Hierarchies

An HPSS storage hierarchy defines the storage classes on which files in that hierarchy are to be stored. A hierarchy consists of multiple levels of storage, with each level representing a different storage class. Files are moved up and down the hierarchy via migrate and stage operations based on usage patterns, storage availability, and site policies. For example, a storage hierarchy might consist of a fast disk, followed by a fast data transfer and medium storage capacity robot tape system, which in turn is followed by a large data storage capacity but relatively slow data transfer tape robot system. Files are placed on a particular level in the hierarchy depending upon the migration levels that are associated with each level in the hierarchy. Multiple copies are controlled by this mechanism. Also data can be placed at higher levels in the hierarchy by staging operations. The staging and migrating of data is shown in File migration and stage operations.

Class of Service (COS)

Each bitfile has an attribute called Class Of Service. The COS defines a set of parameters associated with operational and performance characteristics of a bitfile. The COS results in the bitfile being stored in a storage hierarchy suitable for its anticipated and actual size and usage characteristics. See: Class of Service/hierarchy/storage class.

shows the relationship between COS, storage hierarchies, and storage classes.

User-defined Attributes (UDAs)

User-defined Attributes in HPSS are client-created attributes containing additional metadata. UDAs can be associated with any name space object. UDAs are represented internally as a well-formed XML document. The XML document size limit is 2 GB. The maximum amount of data that can be returned at any one time is configurable via Core Server and API settings.

5.3.2. HPSS servers

HPSS servers include the Core Server, Migration/Purge Server, Gatekeeper, Physical Volume Library, Physical Volume Repository, Mover, Storage System Management System Manager, and Startup Daemon. See: HPSS components.

provides a simplified view of the HPSS system. Each major server component is shown, along with the basic control communication paths (thin arrowed lines). Infrastructure items (those components that "glue together" the distributed servers) are shown at the top of the cube. These infrastructure items are discussed in HPSS infrastructure.

HPSS user interfaces (the clients listed in the figure) are also discussed in HPSS user interfaces.

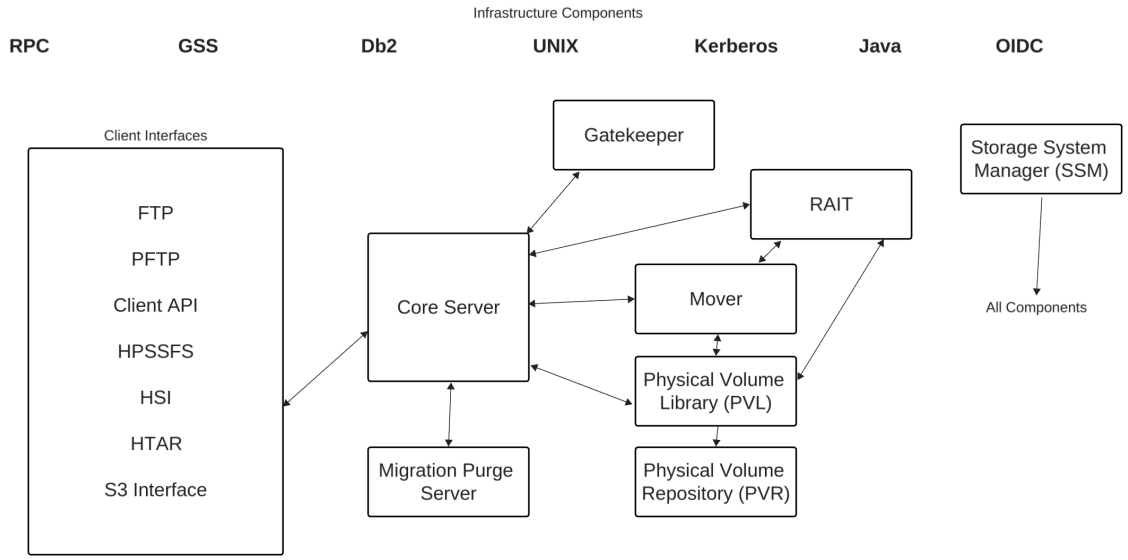


Figure 3. HPSS components

Core Server (CS)

The Core Server provides several key sets of functionality.

First, the Core Server provides translation between human-oriented names and HPSS object identifiers. Name space objects managed by the Core Server are filesets, junctions, directories, files, hard links, and symbolic links. The Core Server provides access verification to objects and mechanisms for manipulating access to these objects via a Portable Operating System Interface (POSIX) view of the name space. This name space is a hierarchical structure consisting of directories, files, and links. These name space objects may exist within filesets that are connected via junctions.

Second, the Core Server provides the abstraction of logical bitfiles to its clients. A bitfile is identified by a Core Server generated name called a bitfile ID. Clients may reference portions of a bitfile by specifying the bitfile ID and a starting address and length. The Core Server supports

random access to files and sparsely written files. It supports parallel reading and writing of data to bitfiles and performs the mapping of logical portions of bitfiles onto physical storage devices. The Core Server supports the migration, purging, and staging of data in a storage hierarchy (though the migration/purge policies are implemented through the Migration/Purge Server, a client to the Core Server).

Third, the Core Server provides a hierarchy of storage objects: storage segments, virtual volumes, and physical volumes. The Core Server translates storage segment references into virtual volume references and then into physical volume references, handles the mapping of physical resources into striped virtual volumes to allow parallel I/O to that set of resources, and schedules the mounting and dismounting of removable media through the Physical Volume Library (see below).
Finally, the Core Server acts as an information clearinghouse to its clients through the HPSS Client API to enable them to locate servers and gather information from both local and remote HPSS systems. It allows a client to determine a server's location and, by knowing other information about the server such as its object UUID, determine its server type or its subsystem ID. This allows a client to contact the appropriate server.

Migration/Purge Server (MPS)

The MPS allows a site to implement its storage management policies by managing the placement of data on HPSS storage media using site-defined migration and purge policies. By making appropriate calls to its Core Server, an MPS copies data to lower levels in the hierarchy (migration), removes data from the current level once copies have been made (purge), or moves data between volumes at the same level (lateral move). Based on the hierarchy configuration, MPS can be directed to create duplicate copies of data when it is being migrated from disk or tape. This is done by copying the data to multiple lower levels in the storage hierarchy.

There are two types of migration: disk migration and tape file migration. The designation disk or tape refers to the type of storage class that migration is running against. See Migration/Purge Server for a more complete discussion of the different types of migration.

MPS runs migration on each storage class periodically using the time interval specified in the migration policy for that class. See HPSS policy modules for details on migration and purge policies. Migration runs can be started automatically when the warning or critical space thresholds for the storage class are exceeded. In addition, migration runs can be started manually by an administrator.

Purge runs are started automatically on each storage class when the free space in that class falls below the percentage specified in the purge policy. Purge runs may also be started manually.

Disk migration/purge:

The purpose of disk migration is to make one or more copies of disk files to lower levels in the hierarchy. The number of copies depends on the configuration of the hierarchy. For disk, migration and purge are separate operations. It is common for disk storage classes which have been configured for migration to also be configured for purge as well. Once a file has been migrated (copied) downwards in the hierarchy, it becomes eligible for purge, which subsequently removes the file from the higher level and allows the disk space to be reused.

Tape file migration:

The purpose of tape file migration is to make an additional copy (or multiple additional copies) of a file, in a tape storage class, to a lower level in the hierarchy. It is also possible to move files downwards instead of copying them. In this case, there is no duplicate copy maintained. There is no separate purge component to tape file migration. Empty volumes must be reclaimed using the reclaim(7) utility.

Gatekeeper (GK)

The Gatekeeper provides two main services:

- It provides sites with the ability to schedule the use of HPSS resources using the Gatekeeping Service.
- It provides sites with the ability to validate user accounts using the Account Validation Service.

Both of these services allow sites to implement their own policy.

The default Gatekeeping Service policy is to not do any gatekeeping. Sites may choose to implement a policy for monitoring authorized callers, creates, opens, and stages. The Core Server will call the appropriate GK API depending on the requests that the site-implemented policy is monitoring.

The Account Validation Service performs authorizations of user storage charges. A site may perform no authorization, default authorization, or site-customized authorization depending on how the accounting policy is set up and whether or not a site has written site-specific account validation code. Clients call this service when creating files, changing file ownership, or changing accounting information. If account validation is enabled, the Account Validation Service determines if the user is allowed to use a specific account or gives the user an account to use, if needed. The Core Server also calls this service to perform an authorization check just before account-sensitive operations take place.

Physical Volume Library (PVL)

The PVL manages all HPSS physical volumes. It is in charge of mounting and dismounting sets of physical volumes, allocating drive and cartridge resources to satisfy mount and dismount requests, providing a mapping of physical volume to cartridge and of cartridge to Physical Volume Repository (PVR), and issuing commands to PVRs to perform physical mount and dismount actions. A primary function of the PVL is the support for atomic mounts of sets of cartridges for parallel access to data. Atomic mounts are implemented by the PVL, which waits until all necessary cartridge resources for a request are available before issuing mount commands to the PVRs.

Physical Volume Repository (PVR)

PVRs manage HPSS cartridges. Though an HPSS system may contain multiple PVRs, each cartridge is managed by only one. PVRs provide APIs for clients to request cartridge mounts and dismounts and query the status of cartridges. For convenience, PVRs are often configured in one-to-one correspondence to tape libraries.

For information on the types of tape libraries supported by HPSS PVRs, see Robotically mounted

tape.

An Operator PVR is provided for cartridges not under control of a robotic library. These cartridges are mounted on a set of drives by operators.

Mover (MVR)

The purpose of the Mover is to transfer data from a source device to a sink device. A device can be a standard I/O device with geometry (such as tape or disk) or a device without geometry (such as network or memory). The Mover's client (typically the Core Server) describes the data to be moved and where the data is to be sent. It is the Mover's responsibility to actually transfer the data, retrying failed requests and attempting to optimize transfers. The Mover supports transfers for disk devices, tape devices and a Mover protocol that can be used as a lightweight coordination and flow control mechanism for large transfers.

Storage System Management System Manager (SSMSM)

SSM, the Storage System Management subsystem, is the tool used by the system administrator to manage HPSS. SSM has three components, one of which is an HPSS server and two of which are user interface client programs. The server is:

SSM System Manager (SSMSM, or hpss_ssmsm)

Communicates with all other HPSS components requiring monitoring or control.

The user interface clients are:

SSM GUI (hpssgui)

Provides the HPSS administrator or operator the ability to configure or monitor the HPSS System through a graphical user interface.

SSM Command-Line Interface (hpssadm)

Provides the HPSS administrator or operator the ability to configure or monitor a subset of the HPSS system through a set of interactive or batch commands.

SSM API

Provides the HPSS administrator or operator the ability to configure or monitor the HPSS system through a REST-based interface.

Web SSM UI

Provides HPSS operators the ability to view current SSM statistics in their browser through a convenient web interface.

SSM enables the administrator to configure, monitor and control the resources (servers, devices, tape libraries, and media) of HPSS in ways that conform to the management policies of a given customer site.

Monitoring capabilities include the ability to query the values of important management attributes of storage system resources and the ability to receive notifications of alarms and other significant system events. Controlling capabilities include the ability to start up and shut down servers and the ability to set the values of management attributes of storage system resources and storage system policy parameters. Additionally, SSM can request that specific operations be performed on resources within the storage system, such as adding and deleting logical or physical resources.

Startup Daemon (SUD)

The Startup Daemon is called by the SSMSM to start each HPSS server except the SSMSM, the Startup Daemon itself, and the remote portion of the Mover. A Startup Daemon is required on each node where any HPSS Server executes, with the exception that no Startup Daemon is required on nodes where the only HPSS Server is the remote portion of the Mover.

5.3.3. HPSS storage subsystems

The goal of storage subsystems (or just "subsystems") is to increase the scalability of HPSS by allowing multiple Core Servers to be used within a single HPSS system. Every HPSS system is partitioned into one or more subsystems. Each subsystem contains a single Core Server. If migration and purge are needed, then the subsystem should contain a single Migration/Purge Server. Each Core Server and each Migration/Purge Server must exist within a storage subsystem. Each subsystem may optionally be serviced by a Gatekeeper which performs site-specific user-level scheduling of HPSS storage requests or account validation. Each Gatekeeper may service multiple subsystems. All other servers exist independently of storage subsystems. Sites which do not need multiple Core Servers use a single storage subsystem.

The computer that runs the Core Server for subsystem X is referred to as the "Subsystem X node" while the computer running the Root Core Server is known as the "Root Subsystem Node".

Each HPSS system consists of two types of DB2 databases. The global database contains subsystem-independent data, and a subsystem database contains subsystem-dependent data. An HPSS system has exactly one global database and one or more subsystem databases.

The definitions of classes of service, hierarchies, and storage classes apply to the entire HPSS system (they are subsystem-independent). All classes of service, hierarchies, and storage classes are known to all storage subsystems within HPSS. The level of resources dedicated to these entities by each storage subsystem may differ. It is possible to disable selected classes of service within given storage subsystems. Although the class of service definitions are global, if a class of service is disabled within a storage subsystem then the Core Server in that subsystem never selects that class of service.

Since the Migration/Purge Server (MPS) is contained within the storage subsystem, migration and purge operate independently in each subsystem. Each Migration/Purge Server is responsible for migration and purge for those storage class resources contained within its particular storage subsystem. Migration and purge runs are independent and are not synchronized. Migration and purge for a storage class may be configured differently for each storage subsystem. It is possible to set up a single migration or purge policy which applies to a storage class across all storage subsystems (to make configuration easier), but it is also possible to control migration and purge differently in each storage subsystem.

Similarly, storage class thresholds may be configured differently for each storage subsystem. It is possible to set up a single set of thresholds which apply to a storage class across all storage subsystems, but it is also possible to control the thresholds differently for each storage subsystem.

5.3.4. HPSS infrastructure

The HPSS infrastructure items (see HPSS components) are those components and services used by

the various HPSS servers. The HPSS infrastructure components common among servers are discussed below.

Remote Procedure Calls (RPC)

Most HPSS servers, with the exception of the MVR, PFTPD, and logging services (see below), communicate requests and status (control information) via RPCs. HPSS does not use RPCs to move user data. RPCs provide a communication interface resembling simple, local procedure calls.

Thread Services

HPSS uses a threads package for multitasking. The threads package is vital for HPSS to serve large numbers of concurrent users and to enable multiprocessing of its servers.

Transaction Management

Requests to perform actions, such as creating bitfiles or accessing file data, result in client-server interactions between software components. The HPSS Core Server performs most of the HPSS metadata changes using the transaction management tools provided by DB2. For the most part, these metadata transactions are managed entirely within the Core Server. Other servers such as MPS and PVL modify their metadata transactionally, and those transactions are entirely contained within those servers. A very small number of rarely performed operations require distributed transaction management, and these are handled by DB2 as well.

Transactional integrity to guarantee consistency of server state and metadata is required in HPSS in case a particular component fails. HPSS metadata updates utilize the transactional capability of DB2. The selection of DB2 was based on functionality and vendor platform support. It provides HPSS with an environment in which a job or action completes successfully or is aborted completely.

DB2 provides a full suite of recovery options for metadata transactions. Recovery of the database to a consistent state after a failure of HPSS or DB2 is automatic. A full suite of database backup and maintenance tools is provided as well.

Security

HPSS security software provides mechanisms that allow HPSS components to communicate in an authenticated manner, to authorize access to HPSS objects, to enforce access control on HPSS objects, and to issue log records for security-related events. The security components of HPSS provide authentication, authorization, enforcement, and audit capabilities for the HPSS components. Customer sites may use the default security policy delivered with HPSS or define their own security policy by implementing their own version of the security policy module.

- ***Authentication*** is responsible for guaranteeing that a principal (a customer identity) is the entity that is claimed, and that information received from an entity is from that entity.
- ***Authorization*** is responsible for enabling an authenticated entity access to an allowed set of resources and objects. Authorization enables end-user access to HPSS directories and bitfiles.
- ***Enforcement*** is responsible for guaranteeing that operations are restricted to the authorized set of operations.
- ***Audit*** is responsible for generating a log of security-relevant activity. HPSS audit capabilities

allow sites to monitor HPSS authentication, authorization, and file security events. File security events include file creation, deletion, opening for I/O, and attribute modification operations.

HPSS components that communicate with each other maintain a joint security context. The security context for both sides of the communication contains identity and authorization information for the peer principals as well as an optional encryption key.

Access to HPSS server interfaces is controlled through an Access Control List (ACL) mechanism. Membership on this ACL is controlled by the HPSS administrator.

Logging

A logging infrastructure component in HPSS provides an audit trail of server events. Logged data may include alarms, events, requests, security audit records, info records, trace information, debug records, and accounting records. HPSS logs are logged to syslog. By default, following running `configure_logging`, Alarms and Event messages in `/var/hpss/log/HPSS.log` will be reflected to the SSM for display there. Log retention and archiving are accomplished using standard UNIX tools such as **logrotate**, along with HPSS tools like hpss_log_archive(7). See Configuring syslog, Recommendations for syslog configuration and Logging overview for more information.

Accounting

The HPSS accounting system provides the means to collect usage information in order to allow a particular site to charge its users for the use of HPSS resources. It is the responsibility of the individual site to sort and use this information for subsequent billing based on site-specific charging policies. For more information on the HPSS accounting policy, refer to HPSS policy modules.

OpenID Connect (OIDC)

HPSS web services like SSM API authenticate using an OIDC provider which mirrors or integrates with HPSS security for issuing authentication tokens.

License Files

HPSS requires a set of license files be installed (see Install HPSS License). These license files confirm that the system is eligible to run HPSS. HPSS will begin to log warnings when the license file is nearing expiration, and administrators can check the validity and settings on the license file using the hpss_license(7) tool.

5.3.5. HPSS user interfaces

As indicated in HPSS components, HPSS provides the user with a number of transfer interfaces as discussed below.

File Transfer Protocol (FTP)

HPSS provides an industry-standard FTP user interface. Because standard FTP is a serial interface, data sent to a user is received serially. This does not mean that the data within HPSS is not stored and retrieved in parallel; it means that the PFTP daemon within HPSS must consolidate its internal parallel transfers into a serial data transfer to the user. HPSS FTP

performance in many cases will be limited not by the speed of a single storage device but by the speed of the data path between the HPSS PFTP daemon and the user's FTP client.

Parallel FTP (PFTP)

The PFTP supports standard FTP commands plus extensions and is built to optimize performance for storing and retrieving files from HPSS by allowing data to be transferred in parallel across the network media. The parallel client interfaces have a syntax similar to FTP but with numerous extensions to allow the user to transfer data to and from HPSS across parallel communication interfaces established between the PFTP client and the HPSS Movers. This provides the potential for using multiple client nodes as well as multiple server nodes. PFTP supports transfers via TCP/IP. The PFTP client establishes a **control** connection with the HPSS PFTP daemon and subsequently establishes TCP/IP **data** connections directly with HPSS Movers to transfer data at rates limited only by the underlying media, communications hardware, and software.

Client Application Program Interface (Client API)

The Client API is an HPSS-specific programming interface that mirrors the POSIX.1 specification where possible to provide ease of use to POSIX application programmers. Additional APIs are also provided to allow the programmer to take advantage of the specific features provided by HPSS (for example, storage/access hints passed on file creation and parallel data transfers). The Client API is a programming-level interface. It supports file open/create and close operations; file data and attribute access operations; file name operations; directory creation, deletion, and access operations; and working directory operations. HPSS users interested in taking advantage of parallel I/O capabilities in HPSS can add Client API calls to their applications to utilize parallel I/O. For the specific details of this interface see the *HPSS Programmer's Reference*.

HPSSFS-FUSE Interface

The HPSSFS-FUSE Interface presents a standard POSIX I/O interface to a user application. This obviates the need for a user application to be rewritten against the HPSS Client API and hence can be used "out of the box" as long as the user application is POSIX-compliant. A portion of an HPSS directory tree can be mounted on a client machine as if it were a local POSIX-compliant file system. See hpssfs(7) and the *HPSSFS-FUSE Administrator's Guide* bundled with HPSSFS-FUSE for more information.

5.3.6. HPSS management interfaces

HPSS provides a graphical user interface, the SSM hpssgui(7), for HPSS administration and operations GUI. The hpssgui(7) simplifies the management of HPSS by organizing a broad range of technical data into a series of easy-to-read graphic displays. The hpssgui(7) allows monitoring and control of virtually all HPSS processes and resources from windows that can easily be added, deleted, moved, or overlapped as desired.

HPSS also provides a command-line SSM interface, hpssadm(7). This tool does not provide all the functionality of the hpssgui(7), but does implement a subset of its frequently used features, such as some monitoring and some control of servers, devices, storage classes, volumes, and alarms. It is useful for performing HPSS administration from remote locations where network traffic is slow or difficult. Additionally, hpssadm(7) provides some rudimentary mass configuration support by means of the ability to issue configuration commands from a batch script.

HPSS also provides a REST-based SSM interface for programmatic access, the SSM API. This provides a simple interface for scripting batch operations and retrieving HPSS status for external processing.

The HPSS SSM is a server which has several clients. These clients fulfill different use cases for system administration and management.

HPSS also provides a Web-based SSM interface, webssm. This interface provides a graphical monitoring dashboard. WebSSM is built on top of SSMAPI.



The hpssgui(7) and hpssadm(7) interfaces are slated to be removed and replaced by the WebSSM and SSMAPI in the future. The WebSSM and SSMAPI both currently provide a limited set of functionality compared to the existing interfaces.

In addition to SSM, HPSS provides a number of command-line utilities for specialized management purposes, such as listing the volumes managed by a particular PVR or core server. See the Management tools section for more information. See also the HPSS man pages for descriptions of these utilities.

5.3.7. HPSS policy modules

There are a number of aspects of storage management that probably will differ at each HPSS site. For instance, sites typically have their own guidelines or policies covering the implementation of accounting, security, and other storage management operations. In order to accommodate site-specific policies, HPSS has implemented flexible interfaces to its servers to allow local sites the freedom to tailor management operations to meet their particular needs.

HPSS policies are implemented using two different approaches. Under the first approach, used for migration, purge, and logging policies, sites are provided with a large number of parameters that may be used to implement local policy. Under the second approach, HPSS communicates information through a well-defined interface to a policy software module that can be completely replaced by a site. Under both approaches, HPSS provides a default policy set for users.

Migration policy

The migration policy defines the conditions under which data is copied from one level in a storage hierarchy to one or more lower levels. Each storage class that is to have data copied from that storage class to a lower level in the hierarchy has a migration policy associated with it. The MPS uses this policy to control when files are copied and how much data is copied from the storage class in a given migration run. Migration runs are started automatically by the MPS based upon parameters in the migration policy.

Note that the number of copies which migration makes and the location of these copies is determined by the definition of the storage hierarchy and not by the migration policy.

Purge policy

The purge policy defines the conditions under which data that has already been migrated from a disk storage class can be deleted. Purge applies only to disk storage classes. It is common, but not necessary, for disk storage classes which have a migration policy to also have a purge policy. Purge runs are started automatically by the MPS based upon parameters in the purge policy.

Logging policy

The logging policy controls the types of messages to log. On a per-server basis, the message types to write to the HPSS log may be defined. In addition, for each server, options to send Alarm and Event messages to SSM may be defined.

Security policy

Security policy defines the authorization and access controls to be used for client access to HPSS. HPSS security policies are provided to control access (authentication) from FTP or Parallel FTP (or both) using **Username/Password** or **Kerberos** credentials.

HPSS provides facilities for recording information about authentication and object (file and directory) creation, deletion, access, and authorization events. The security audit policy for each server determines the records that each individual server will generate. All servers can generate authentication records.

Accounting policy

The accounting policy provides runtime information to the accounting report utility and to the account validation service of the Gatekeeper. It helps determine what style of accounting should be used and what level of validation should be enforced.

The two types of accounting are site-style and UNIX-style. The site-style approach is the traditional type of accounting in use by most mass storage systems. Each site will have a site-specific table (Account Map) that correlates the HPSS account index number with their local account charge codes. The UNIX-style approach allows a site to use the User Identifier (UID) for the account index. The UID is passed along in UNIX-style accounting just as the account index number is passed along in site-style accounting.

Account validation allows a site to perform usage authorization of an account for a user. It is turned on by enabling the Account Validation field of the *Accounting Policy* configuration screen. If account validation is enabled, the accounting style in use at the site is determined by the Accounting Style field. A site policy module may be implemented by the local site to perform customized account validation operations. The default account validation behavior is performed for any account validation operation that is not overridden by the site policy module.

Gatekeeping policy

The Gatekeeper provides a gatekeeping service along with an account validation service. These services provide the mechanism for HPSS to communicate information though a well-defined interface to a policy software module that can be written by a site. The site policy code is placed in well-defined shared libraries for the gatekeeping policy and the accounting policy (`libgksite.[a|so]` for the gatekeeping policy and `libacctsite.[a|so]` for accounting) which are linked to the Gatekeeper. The Gatekeeper looks for these libraries in `/usr/local/lib64` first, and then `/opt/hpss/lib`. The gatekeeping policy shared library contains a default policy which does *no* gatekeeping. Sites will need to enhance this library to implement local policy rules if they wish to monitor and load-balance requests.

5.4. HPSS hardware platforms

5.4.1. Server platforms

HPSS requires at least one Linux node for the core server components. A server node must have sufficient processing power and memory to handle the workload.

5.4.2. Client platforms

The full-function Client API can be ported to any platform that supports UNIX.

The PFTP client code and Client API source code for platforms other than Linux are not on the HPSS distribution image. Maintenance of the PFTP and Client API software on platforms other than Linux is the responsibility of the customer, unless a support agreement is negotiated with IBM. Contact HPSS support for information on how to obtain the needed software.

The following matrix illustrates which platforms support HPSS interfaces.

Table 1. HPSS client interface and Mover platforms

Platform	**HPSS Mover**	**PFTP client**	**Client API**	**FTP clients** *(see Note 1)*	**SSM API**
RHEL (x86)	X	X	X	X	X
RHEL (arm)		X	X	X	
RHEL (Power PC)	X	X	X	X	X

Note 1: GUI-based clients may not function correctly for some commands.
Note 2: For compatibility of HPSS applications such as HSI/HTAR, HPSSFS-FUSE, and others, consult application documentation.

5.4.3. Mover platforms

Movers are used to control the logical network attachment of storage devices and are configured to run on one or more nodes. A Mover consists of two parts: The Mover administrative process that runs on the server node, and the remote Mover process that handles the HPSS devices and data transfers. See HPSS client interface and Mover platforms above for a detailed list of supported platforms.

Chapter 6. HPSS planning

6.1. Overview

This chapter provides HPSS planning guidelines and considerations to help the administrator effectively plan, and make key decisions about, an HPSS system.

The planning process for HPSS must be done carefully to ensure that the resulting system satisfies the site's requirements and operates in an efficient manner. The administrator should read this entire chapter before planning the system.

The following paragraphs describe the recommended planning steps for the HPSS installation, configuration, and operational phases.

6.1.1. HPSS system architecture

Before getting into the details of storage sizing, it is best to understand the overall HPSS system and how the components are arranged to build the HSM. The following illustration shows the basic architecture of an HPSS system including disk and tape resources and their relationship to HPSS server nodes, Mover nodes, internal and external networks, and SAN interconnections. Specifics of this architecture for a given site are developed during the proposal and initial project planning stages of a deployment. Ideally, the required space is derived from requirements gathered from the HPSS Questionnaire document, known operational constraints, transfer and transaction rates, and anticipated storage capacity needs. Often the disk and tape resources are dictated by current equipment already available and budgetary constraints on what can be purchased. Specific quantities and sizing of these resource are beyond the scope of this planning document. These are largely defined by the above inputs and negotiations during the initial planning meeting in which the systems engineering team draws from experience and similar deployments to design a working architecture that balances the end-user requirements with the potential, or actual, resources available.

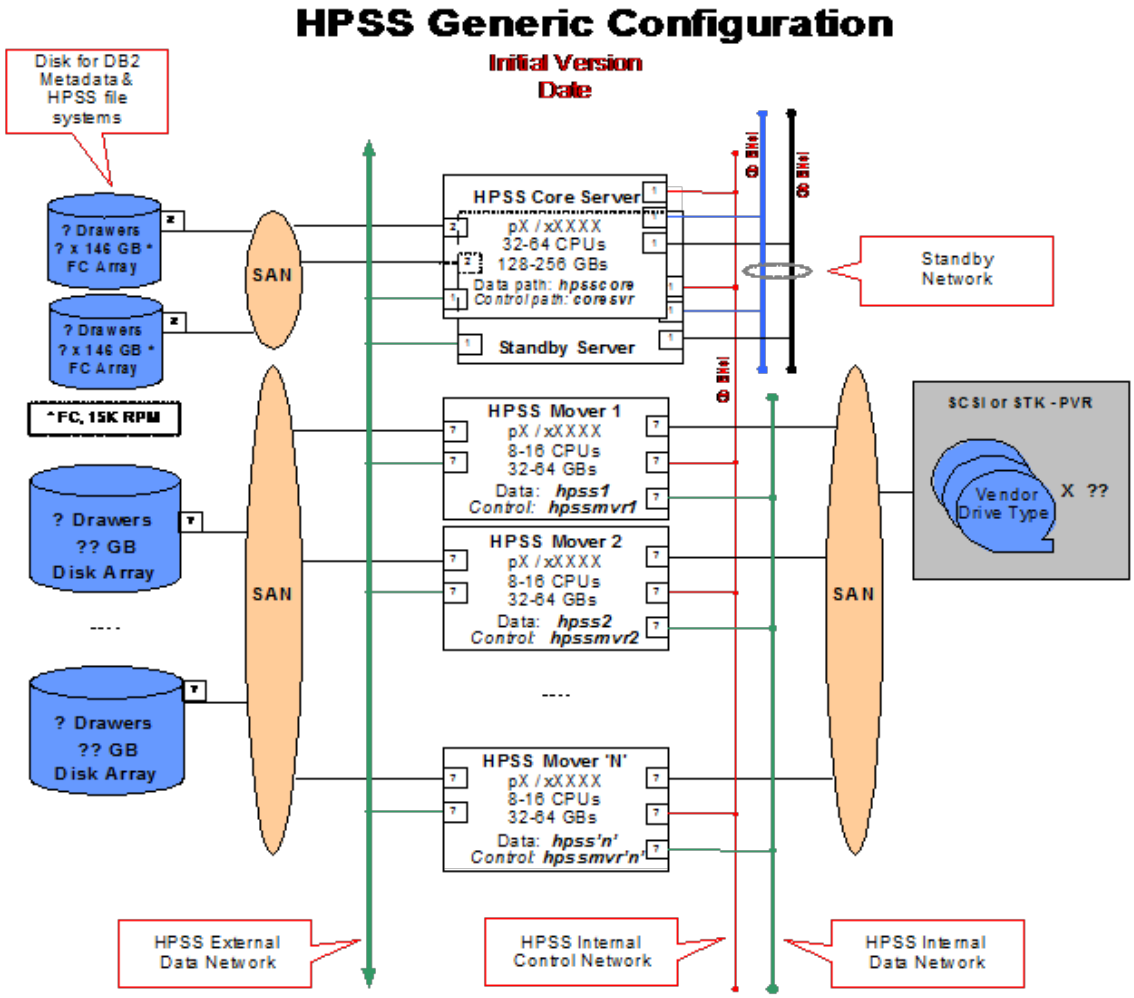


Figure 4. HPSS generic configuration

6.1.2. HPSS configuration planning

Before beginning the planning process, there is an important issue to consider. HPSS was designed to optimize the transfer of large files at the expense of some small file transfer performance. If at all possible, try to reduce the number of small files that are introduced into your HPSS system. For example, if you plan to use HPSS to backup all of the PCs in your organization, it would be best to aggregate the individual files into large individual files before moving them into the HPSS name space.

The following planning steps must be carefully considered for the HPSS infrastructure configuration and the HPSS configuration phases:

1. Identify the site's storage requirements and policies, such as the initial storage system size, anticipated growth, usage trends, average file size, expected throughput, backup policy, and availability. For more information, see Requirements and intended uses for HPSS.
2. Define the architecture of the entire HPSS system to satisfy the above requirements. The planning should:
   - Identify the nodes to be configured as part of the HPSS system.
   - Identify the type of network that will be used for the HPSS system. HPSS allows the administrator to configure the system to use IPv4-only (the default), IPv6 mixed-mode (where IPv6 is preferred), and IPv6-only mode.

Note: The HPSS STK PVR is not supported on IPv6 due to the Oracle StorageTek CDK ACSAPI library not supporting it at this time. If you require this support, consult your Oracle representative about implementing this enhancement.

- Identify the disk and tape storage devices to be configured as part of the HPSS system and the nodes and networks to which each of the devices will be attached. Storage devices can be assigned to a number of nodes to allow data transfers to utilize the devices in parallel without being constrained by the resources of a single node. This capability also allows the administrator to configure the HPSS system to match the device performance with the performance of the network used to transfer the data between the HPSS Movers and the end users (or other HPSS Movers in the case of internal HPSS data movement for migration and staging). Refer to Hardware considerations for more discussions on the storage devices and networks supported by HPSS.
- Identify the HPSS subsystems to be configured and how resources will be allocated among them. Refer to Storage subsystem considerations for more discussion on subsystems.
- Identify the HPSS servers to be configured and the node where each of the servers will run. Refer to HPSS server considerations for more discussions on the HPSS server configuration.
- Identify the HPSS user interfaces (such as FTP, PFTP, or Client API) to be configured and the nodes where the components of each user interface will run. Refer to HPSS interface considerations for more discussion on the user interfaces supported by HPSS.

3. Ensure that the prerequisite software has been obtained, installed, and configured properly in order to satisfy the target HPSS architecture. Refer to Prerequisite software considerations for more information on the HPSS prerequisite software requirements.
4. Identify the OIDC provider that will be used with HPSS web services such as Web SSM and SSM API. Obtain certificates from a trusted certificate authority that can be installed as part of web server configuration. Identify how these certificates will be refreshed, and who will manage OIDC provider access over time.
5. Plan how to manage logging from HPSS web services - whether to archive logging and where to archive it to. How many on disk backups to keep. How long to retain web service logging.
6. Determine the DB2 disk storage space needed to satisfy the requirements of the HPSS system, and verify there is sufficient free space in the file systems to meet those needs. Refer to HPSS metadata space for more discussion of file system storage requirements for HPSS and DB2.
7. Verify that each of the identified nodes has sufficient resources to handle the workloads to be imposed on the node. Refer to System memory and disk space for more discussions on the system resource requirements.
8. Plan the design of the HPSS storage characteristics and HPSS storage space to satisfy the site's requirements:
   - Plan for file families, if any. Refer to File families for more information about configuring families.
   - Plan for filesets and junctions, if any. Refer to the Filesets and junctions chapter for more information.
   - Plan for HPSS storage classes. Refer to Storage class for more information on the storage class configuration.

- Plan for HPSS storage hierarchies. Refer to Storage hierarchy for more information on the storage hierarchy configuration.
- Plan for HPSS classes of service. Refer to Class of Service for more information on the Class of Service configuration.
- Plan the migration and purge policy for each storage class. Refer to Migration policy and Purge policy for more information.
- Determine the amount of user data storage space needed for each storage class. Refer to HPSS user storage space for more information on the HPSS storage space considerations.
- Identify the disk and tape media to be imported into HPSS.

9. Define the accounting policy to be used. Refer to Accounting policy and validation for more information on the accounting policy configuration.
10. Define the logging policy for each of the HPSS servers. Refer to Logging policy for more information on the logging policy configuration.
11. Define the security policy for the HPSS system. Refer to Security policy for more information on the security policy for HPSS.
12. Determine if a Gatekeeper will be required. It is required if a site wants to do account validation or gatekeeping. Refer to Accounting policy and validation and Gatekeeping for more information.
13. Identify the site's need for User-defined Attributes, if any. Planning should include determining the amount of DB2 disk storage space needed to satisfy their UDA requirements, deciding which attributes should have indexes, and developing an XML schema if desired.

6.1.3. Purchasing hardware and software

It is recommended that hardware be purchased only after the HPSS configuration has been planned. Purchasing the hardware prior to the planning process may result in performance and utilization issues that could easily be avoided by advance planning.

When purchasing Linux servers for storage purposes, note that the operating system limitations will only allow a fixed number of raw devices to be configured per logical unit (disk drive or disk array). Linux operating system limits SCSI disks to 15 partitions and limits IDE disks to 63 partitions, unless LVM is used. These limits can potentially impact the utilization of a disk drive or disk array.

Refer to HPSS sizing considerations for more information on calculating the number and size of devices that will be needed to meet your requirements.

Refer to Prerequisite software considerations for more information on the HPSS prerequisite software requirements.

6.1.4. HPSS operational planning

The following planning steps must be carefully considered for the HPSS operational phase:

1. Define the site guidelines for the HPSS users and SSM users.

- Each HPSS user who uses the storage services provided by HPSS should be assigned an Accounting ID and one or more appropriate Classes of Service (COS) to store files.
- Each SSM user (administrator or operator) should be assigned an appropriate SSM security level. The SSM security level defines what functions each SSM user can perform on HPSS through SSM. Refer to the SSM user security section, the Creating the SSM user accounts section, and the Add an SSM user ID section for more information on setting up the security level for an SSM user. For sites using the SSM API or Web SSM, the SSM users must be created in the OIDC solution with the same name as their HPSS user ID.

2. Define the site procedures for repacking and reclaiming HPSS tape volumes. Define the tape consolidation and reuse goals. For instance, define a tape utilization factor and plan to repack those tapes that fall below that limit. The limits can be set on a per storage class basis. Also, decide when or if empty tapes will be reclaimed. Refer to the Repack Virtual Volumes window section and the Reclaim Virtual Volumes window section for more information.
3. Define the site policy and procedure for generating the accounting reports. Take into consideration how often an accounting report needs to be generated, how the accounting information from the report will be used to produce the desired cost accounting, and whether the accounting reports need to be archived. Refer to Accounting policy and validation for more information on defining an accounting policy and generating accounting reports.
4. Determine if gatekeeping (monitoring or load-balancing) will be required. If so, define and write the site policy code for gatekeeping. Refer to Gatekeeping for more information on gatekeeping, and refer to the *HPSS Programmers Reference* for guidelines on implementing the Site Interfaces for the Gatekeeping Service.
5. Determine the desired site trashcan settings. Take into consideration the number of threads that will be devoted to the incineration process, how often the incinerator will run, the amount of time needed before a file becomes eligible for incineration, and how often the statistics thread will run. Refer to the HPSS trashcans section for more information on trashcan settings.

6.1.5. HPSS deployment planning

The successful deployment of an HPSS installation is a complicated task which requires reviewing client and system requirements, integration of numerous products and resources, proper training of users and administrators, and extensive integration testing in the client environment.

To help the HPSS system administrators in all of these tasks, a set of procedures have been developed to supplement this document. The HPSS Deployment Process is a document maintained by the HPSS deployment team and contains a detailed outline of what is required to bring up an HPSS system, from an initial introduction and review of the environment to production use. This document is provided to customers at the initial HPSS planning meeting. The deployment procedures include a timeline plus checklist that the HPSS customer installation/system administration team should use to keep the deployment of an HPSS system on track. This is the same guide that HPSS support uses to monitor and check the progress of an installation.

6.2. Requirements and intended uses for HPSS

This section provides some guidance for the administrator to identify the site's requirements and expectations of HPSS. Issues such as the amount of storage needed, access speed and data transfer

speed, typical usage, security, expected growth, data backup, and conversion from an old system must be factored into the planning of a new HPSS system.

6.2.1. Storage system capacity

The amount of HPSS user data storage space the administrator must plan for includes the following considerations:

- The amount of user data storage space required to support new user accounts.
- The amount of user data storage space required to support expected growth of current accounts.
- The amount of user data storage space required to support storage management activities such as migration and repack.
- The amount of user data storage space required to support duplicate copies of user files.

Another component of storage space planning is the amount of space needed for HPSS system metadata. Refer to HPSS user storage space and HPSS metadata space for more information on determining the needed storage space and metadata space.

6.2.2. Required throughputs

Determine the required or expected throughput for the various types of data transfers that users will perform. Some users want quick access to small amounts of data. Other users have huge amounts of data they want to transfer quickly, but are willing to wait for tape mounts or other delays. In all cases, plan for peak loads that can occur during certain time periods. These findings must be used to determine the type of storage devices and network to be used with HPSS to provide the needed throughput.

6.2.3. Load characterization

Understanding the kind of load users are putting on an existing file storage system provides input that can be used to configure and schedule the HPSS system. What is the distribution of file sizes? How many files and how much data is moved in each category? How does the load vary with time (over a day, a week, a month)? Are any of the data transfer paths saturated?

Having this storage system load information helps to configure HPSS so that it can meet the peak demands. Also based on this information, maintenance activities such as migration, repack, and reclaim can be scheduled during times when HPSS is less busy.

See the section on disk I/O operations metrics for more information about monitoring how I/O is load-balanced across disk movers and VVs in a running HPSS system: HPSS Disk Mover and VV Load-Balancing

6.2.4. Usage trends

To configure the system properly the growth rates of the various categories of storage, as well as the growth rate of the number of files accessed and data moved in the various categories must be known. Extra storage and data transfer hardware must be available if the amount of data storage

and use are growing rapidly.

6.2.5. Duplicate file policy

The policy on duplicating user data files impacts the amount of data stored and the amount of data moved. If all user files are duplicated, the system will require twice as much tape storage. Users can be given control over duplication of their files by allowing them a choice between hierarchies which provide duplication and hierarchies which do not.

6.2.6. Charging policy

HPSS does not charge users for the use of storage system resources. Instead, it collects accounting information which a site can use to implement a charging policy. Even if sites do not charge for HPSS usage, the accounting information should be examined regularly in order to understand the system resource utilization. This information can assist sites in adjusting their system configuration to better handle their workload.

6.2.7. Security

Authentication and authorization between HPSS servers is done through the use of either UNIX or Kerberos security tools for authentication and either UNIX or LDAP for authorization services. By default, servers are authenticated using the Kerberos authentication service, and authorization information is obtained from the UNIX authorization service. The default protection level passes authentication tokens on the first remote procedure call to a server. The authentication service, authorization service, and protection level for each server can be configured to raise or lower the security of the system. Two cautions should be noted:

1. raising the protection level to packet integrity or packet privacy will require additional processing for each RPC, and
2. lowering the authentication service to none effectively removes the HPSS authentication and authorization mechanisms.

Lowering the authentication service level should only be done in a trusted environment.

Each HPSS server authorizes and enforces access to its interfaces through access control lists stored in the AUTHZACL table. To modify the server state, control access is required. Generally, this is given only to the principal associated with the HPSS system administrative component, which by default is hpssssm. (This principal is defined by the HPSS_PRINCIPAL_SSM environment variable.) Additional principals can be allowed or denied access by setting permissions appropriately. See the HPSS server security ACLs section for more information.

Security auditing in each server may be configured to record all, none, or some security events. Some sites may choose to log every client connection; every bitfile creation, deletion, and open; and every file management operation. Other sites may choose to log only errors. See the security information fields in the general server configuration (in the Server configuration section) for more details.

User access to HPSS interfaces depends on the interface being used. Access through the native Client API uses the UNIX or Kerberos authentication services and UNIX or LDAP authorization

services described above. FTP or Parallel FTP access may utilize the HPSS password file, a configurable password file, or the Kerberos credentials. Additional FTP access is available using Kerberos GSS credentials. Refer to the *FTP* section of the *HPSS User's Guide* for additional details.

Web SSM and SSM API access to SSM relies on Open ID Connect (OIDC). This allows access to the SSM interface using limited time tokens. The level of access available to the Web SSM or SSM API still depends on the SSM user role (administrator or operator).

6.2.7.1. Cross realm access

Kerberos provides facilities for secure communication between multiple Kerberos realms (domains) referred to as Trusted "Cross Realm" access. These features use the Kerberos facilities to provide a trusted environment between cooperating locations. HPSS uses the Kerberos Cross Realm features for authentication. The procedures for interconnecting Kerberos realms are outlined in the Security services section. The HPSS Parallel FTP program can utilize the Kerberos and HPSS Cross Realm access features.

The Generic Security Service (GSS) FTP, available from the Massachusetts Institute of Technology, and the HPSS Parallel FTP applications can take advantage of the Cross Realm access features for authentication and authorization (subject to certain caveats: see FTP documentation for details). The pftp_client(7) binary must be built using the distributed source code. However, it is the site's responsibility to obtain the necessary Kerberos components.

ACL entries in the AUTHZACL table and/or ACLs on HPSS directories and files may need to be added for appropriate foreign_user or foreign_group (or both) entries.

6.2.8. HPSS availability options

When configured with the appropriate redundant hardware, HPSS has some add-on options that allow a system to continue operations with a minimal amount of downtime: Automatic HA with Cluster Management Software, Core Server Manual Failover, and Core Server HADR Failover.

Automatic HA with Cluster Management Software

The Automated HA option for HPSS uses cluster management software to provide high availability. With RHEL 6, Red Hat's Cluster Suite Services is used to facilitate HA for HPSS. For RHEL 7, Tivoli System Automation for Multiplatforms (TSA/MP) is used to implement cluster services for the HA HPSS solution.

Core Server Manual Failover

A pair of identically-configured servers with a shared set of metadata and file system resources are used as the core server redundant pair. Only one node is active at any time, running the Core Services and DB2 components. A set of scripts allow the administrator to start and stop service on each node, and the start script prevents the accidental startup of HPSS or DB2 if already active on the other node. This option protects against node loss, but not against the total loss of the shared metadata and file system resource.

Core Server HADR Failover

Like the Core Server Manual failover option, a pair of identically-configured servers are used; however, each node has its own set of metadata and file system resources. In normal

operational mode, a copy of DB2 runs on each node: one acting as the HADR primary copy, the other operating as the HADR secondary copy. The secondary server must be active and ready to receive logs and replay transactions from the primary. A set of scripts are used to start and stop HPSS on the primary copy node, and another set of scripts are used to switch HADR roles between the nodes. This option protects against the loss of one server node or one of the metadata and file system resources.

6.3. Prerequisite software considerations

This section defines the prerequisite requirements for HPSS. Some products must be obtained separately from HPSS and installed prior to the HPSS installation and configuration.

6.3.1. Prerequisite software overview

This section describes the prerequisite software packages required by HPSS and provides information to obtain them. Refer to the *HPSS Release Notes* for specific versions.

6.3.1.1. DB2

HPSS uses DB2 for Linux, UNIX and Windows by IBM Corporation to manage all HPSS metadata. DB2 software is included in the HPSS distribution. Refer to Install DB2 and set up permanent license for more information. The required DB2 FixPak must be downloaded and installed after the DB2 base is installed. DB2 FixPaks can be downloaded from the DB2 for Linux, UNIX, and Windows webpage at http://www-01.ibm.com/support/docview.wss?rs=71&uid=swg27007053.

6.3.1.2. OpenSSL

HPSS uses the OpenSSL Project's cryptographic library for securing UNIX authentication. OpenSSL comes bundled with Linux. Refer to http://www.openssl.org/ for additional information about OpenSSL.

6.3.1.3. Kerberos

HPSS uses the Massachusetts Institute of Technology (MIT) Kerberos to implement Kerberos authentication. MIT Kerberos is a network authentication protocol designed to provide authentication for client/server applications by using secret-key cryptography. A free implementation of this protocol can be downloaded from the MIT website (http://web.mit.edu/kerberos/). Refer to Install MIT Kerberos for more information.

For Linux, Kerberos is installed as part of the operating system.

6.3.1.4. LDAP and IBM Kerberos

HPSS can be configured to use an LDAP directory to store its authorization information such as users' names, UIDs, GIDs, and home directories. The supported LDAP server products for this release are IBM Tivoli Directory Server and OpenLDAP. TDS can be downloaded from IBM.com. OpenLDAP can be downloaded from the OpenLDAP project page.

Installing the IBM Kerberos client is only required if TDS is being used for authorization and the

LDAP daemon will be used for Kerberos authentication.

OpenLDAP is necessary for Kerberos authentication on Linux. OpenLDAP uses MIT Kerberos, so installing IBM Kerberos would be unnecessary for setting up an OpenLDAP server.

For either case, if Kerberos authentication to LDAP is required, OpenLDAP's client API will be required for compilation into HPSS.

6.3.1.5. Java

HPSS uses the Java Standard Edition to implement the SSM graphical user interface, hpssgui(7), the SSM command-line interface, and hpssadm(7).

The Java product required for the RHEL platform can be downloaded from IBM's download webpages:

http://www.ibm.com/developerworks/java/jdk/linux/download.html.

6.3.1.6. Use of libTI-RPC

HPSS uses the transport-independent RPC (TI-RPC) library for server communications. This library is available via standard repositories and installation media for RHEL.

6.3.1.7. Jansson

Some HPSS servers make use of the Jansson library for converting output into the JSON format. This library is available via standard repositories and installation media for RHEL. For all versions of RHEL 8, version 2.11 of the Jansson library is required.

6.3.1.8. STK Tools

For sites which use an STK PVR, a set of tools to enable communication between the STK PVR and the tape robot is needed, called "STK Toolkit". This toolkit will be provided by HPSS support.

6.3.1.9. Apache Commons Codecs

This package implements a needed security function in Java. This is available as an RPM package on RHEL: apache-commons-codec.

6.3.1.10. nginx

HPSS uses the nginx web server software primarily as a reverse proxy and load balancer for the SSM API. This is available via standard repositories and installation media for RHEL.

6.3.1.11. OIDC Provider

HPSS web services such as Web SSM and SSM API require an OIDC (Open ID Connect) provider to be configured to provide authentication services for HPSS web services. Sites must provide and configure their own OIDC provider that supports the OpenID Connect protocol. HPSS does not include or provide an OIDC provider such as Keycloak. For sites that choose to use Keycloak, HPSS provides an optional helper script to assist with initial Keycloak setup.

See Configure OIDC Provider for more information on how to configure your OIDC provider to work with HPSS.

6.3.2. Prerequisite summary By HPSS node type

This section provides a summary list of prerequisite software required for HPSS. It also lists the software versions which have been verified with HPSS.

6.3.2.1. HPSS server nodes

This section describes the prerequisite software required for each server node.

Linux requirements

- *See the release notes for the HPSS release you are using for the latest information on software prerequisites for the Linux server node.*
- Linux Kernel parameter changes:
  - ASLR or Address Space Layout Randomization is a feature activated by default on some of the newer Linux distributions. It is designed to load shared memory objects in random addresses. In DB2, multiple processes map a shared memory object at the same address across the processes. It was found that DB2 cannot guarantee the availability of the address for the shared memory object when ASLR is turned on.
  - Turn off the randomization by setting the following kernel parameter in `/etc/sysctl.conf`: `kernel.randomize_va_space=0`
  - The system must be rebooted for the change to be recognized.

HPSS Mover nodes

A Mover consists of two processes: the Mover administrative process that runs on the server node, and the remote Mover process that handles the HPSS devices and data transfers. To maximize performance, the remote Mover should not be placed on a node with DB2 and HPSS subsystem servers.

Since HPSS security, logging, and metadata services are performed on the server node, no additional software, like DB2 or HPSS servers, need be installed on the remote Mover node.

Linux requirements

See the release notes for the HPSS release you are using for the latest information on software prerequisites for the Linux Mover node.

6.3.2.2. HPSS client nodes

This section describes the prerequisite requirements for running HPSS clients.

SSM client requirements

The client node where the SSM hpssgui(7) and hpssadm(7) applications run must meet the

following requirements:

- Supported platforms: Linux, Windows

Clients using the SSM API must meet the following requirements:

- Support for TLS 1.2 or higher
- HTTP client library (Python requests, Java HttpClient, etc.)

Client API requirements

The client node where HPSS Client API applications run must meet the following requirements:

- For supported platforms, refer to Client platforms for a complete list.

FTP/PFTP client requirements

The client node where HPSS FTP and PFTP run must meet the following requirements:

- For supported platforms, refer to Client platforms for a complete list.

6.4. Hardware considerations

This section describes the hardware infrastructure needed to operate HPSS and includes considerations about infrastructure installation and operation that may impact HPSS.

6.4.1. Network considerations

Because of its distributed nature and high-performance requirements, an HPSS system is highly dependent on the networks providing connectivity among the HPSS servers and clients.

For control communications (that is, all communications except the actual transfer of data) among the HPSS clients and servers, HPSS requires TCP/IP services. Since control requests and replies are relatively small in size, a low-latency network usually is well suited to handling the control path.

The data path is logically separate from the control path and may also be physically separate, although this is not required. For the data path, HPSS supports the same TCP/IP networks as those supported for the control path. For supporting large data transfers, the latency of the network is less important than the overall data throughput.

HPSS provides support for IPv4 and IPv6 networking. By default, HPSS will only utilize IPv4 networks. If IPv6 is required, there are two new options: IPv6 mixed-mode (an IPv4/IPv6 environment that gives preferential treatment to IPv6 addresses) and IPv6-only mode that will only utilize IPv6 addresses.



The HPSS STK PVR is not supported on IPv6 due to the Oracle StorageTek CDK ACSAPI library not supporting it at this time. If you require this support, consult your Oracle representative about implementing this enhancement.

HPSS also supports a special data path option that may indirectly affect network planning because

it may offload or shift some of the networking load. This option uses the shared memory data transfer method, which provides for intra-node transfers between either Movers or Movers and HPSS clients via a shared memory segment.

Along with shared memory, HPSS also supports a Local File Transfer (LFT) data path for client transfers that involve HPSS Movers that have access to the client's file system. In this case, the HPSS Mover can be configured to transfer the data directly to or from the client's file.

6.4.2. Robotically mounted tape

All HPSS PVRs are capable of sharing a library with other tape management systems but care must be taken when allocating drives among multiple library users. If it is necessary to share a drive between HPSS and another tape management system, the drive can be configured in the HPSS PVR but left in the LOCKED state until it is needed. To help the administrator remember why the drive is LOCKED, it is recommended that a text **Comment** be added to the drive metadata via the *PVL Drive Information* window or the *Tape Device Configuration* window to help the administrators recall why the drive is LOCKED; see the Devices and Drives window section for more information about device and drives management. When needed by HPSS, the drive should be set to UNLOCKED and should not be used by any other tape management system while in this state. This is critical because HPSS periodically polls all of its unlocked drives even if they are not currently mounted or in use.

Generally, only one HPSS PVR is required per library. However, it is possible for multiple PVRs to manage a single library in order to provide drive and tape partitions within a library. The drives in the library must be partitioned among the PVRs and no drive should be configured in more than one PVR. Each tape is assigned to exactly one PVR when it is imported into the HPSS system and will only be mounted in drives managed by that PVR.

HPSS supports tape libraries from IBM, Spectralogic, and Oracle.

6.4.2.1. Drive-controlled LTO libraries (IBM, Spectralogic)

IBM and Spectralogic tape libraries and robots must be attached to a Linux workstation through a SCSI interface. In each case, the library shares a SCSI channel with one of the drives, so at least one of the drives in the library must be connected to the workstation. This workstation must be an HPSS node running the PVR. The SCSI PVR is used to communicate with these libraries.

The HPSS SCSI PVR is compatible with libraries which implement T10 SPC-3 and SMC-3 standards, and libraries which support a code set of 2 or 3.

6.4.2.2. Oracle StorageTek

The SCSI PVR may be used, as described above, for Oracle libraries which support a SCSI interface.

6.4.2.3. Oracle StorageTek tape libraries that support ACSLS

The STK PVR must be able to communicate with Oracle StorageTek's ACSLS server. HPSS requires any release of ACSLS version 7 or version 8. For example, the SL8500 supports ACSLS. For the STK PVR to communicate with the ACSLS server, it must have a TCP/IP connection to the server (for example, Ethernet) and the ACSLS Server System Interface (SSI) client software must be running on the node with the PVR. The client software, maintained by Oracle Corp., can be obtained through

HPSS support; see the STK PVR and STK PVR additional information sections for additional information. Multiple Oracle StorageTek libraries can be connected via pass through ports and managed by a single ACSLS server. This collection of libraries can be managed by a single HPSS STK PVR.

6.4.3. Manually mounted tape

An operator PVR is used to manage a homogeneous set of manually mounted drives. Multiple operator PVRs, with different drive types, can be configured without any additional considerations. Tape mount requests will be displayed on an SSM screen.

6.4.4. Tape devices

The tape devices/drives supported by HPSS are listed in the following table.

Table 2. Supported platform/driver/tape drive combinations

Platform	Driver
Devices	Linux
Native	3580 (Gen3, Gen4, Gen5, Gen6, Gen7, Gen8, Gen9, Gen10), 3592 (Gen2, Gen3, Gen4, Gen5, Gen5A, Gen6, Gen7), 9840 (C, D), 9940 (A, B), T10000 (A, B, C, D)

The "Driver" column uses the following abbreviations:

Native Linux native SCSI Tape Device Driver

Older tape drives (3590, 3590E, 3590H, 9840 (A and B), 3580 (Gen1 and Gen2), 3592 (Gen1 and Gen2) will continue to be supported for existing HPSS sites until they can be upgraded.

6.4.4.1. Multiple media support

Certain drive types have the ability to mount multiple media formats. HPSS supports this ability by maintaining a drive preference table in the PVL for any media that can be mounted in multiple drive types. The drive preference table, drive availability, and mount operation are used in combination to determine what drive type to mount a particular media format in.



Idle drives are given priority over busy drives during the decision process.

The table below shows the drive preference for each media format that can be mounted in multiple drive types.

Table 3. Cartridge/drive affinity table

Cartridge type	Drive preference list	Operation supported
Single-Length 3590	Single-Length 3590 Double-Length 3590 Single-Length 3590E Double-Length 3590E Single-Length 3590H Double-Length 3590H	R/W R/W R R R R
Double-Length 3590	Double-Length 3590 Double-Length 3590E Double-Length 3590H	R/W R R
Single-Length 3590E	Single-Length 3590E Double-Length 3590E Double-Length 3590H	R/W R/W R
Double-Length 3590E	Double-Length 3590E Double-Length 3590H	R/W R
Single-Length 3590H	Single-Length 3590H Double-Length 3590H	R/W R/W
Double-Length 3590H	Double-Length 3590H	R/W
3580 (LTO) Gen 1	3580 (LTO) Gen 1 3580 (LTO) Gen 2 3580 (LTO) Gen 3	R/W R/W R
3580 (LTO) Gen 2	3580 (LTO) Gen 2 3580 (LTO) Gen 3 3580 (LTO) Gen 4	R/W R/W R
3580 (LTO) Gen 3	3580 (LTO) Gen 3 3580 (LTO) Gen 4 3580 (LTO) Gen 5	R/W R/W R
3580 (LTO) Gen 4	3580 (LTO) Gen 4 3580 (LTO) Gen 5 3580 (LTO) Gen 6	R/W R/W R
3580 (LTO) Gen 5	3580 (LTO) Gen 5 3580 (LTO) Gen 6 3580 (LTO) Gen 7	R/W R/W R
3580 (LTO) Gen 6	3580 (LTO) Gen 6 3580 (LTO) Gen 7	R/W R/W
3580 (LTO) Gen 7	3580 (LTO) Gen 7 3580 (LTO) Gen 8	R/W R/W
3580 (LTO) Gen 7 M8	3580 (LTO) Gen 8	R/W
3580 (LTO) Gen 8	3580 (LTO) Gen 8 3580 (LTO) Gen 9	R/W R/W
3580 (LTO) Gen 9	3580 (LTO) Gen 9	R/W

Cartridge type	Drive preference list	Operation supported
3580 (LTO) Gen 10 3580 (LTO) Gen 10 Enterprise	3580 (LTO) Gen 10	R/W
3592 J1A JA/JW Tape 3592 J1A JJ/JR Tape	3592 J1A 3592 E05 (TS1120) 3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R/W R R
3592 E05 JA/JW Tape 3592 E05 JB/JX Tape 3592 E05 JJ/JR Tape	3592 E05 (TS1120) 3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R/W R
3592 E06 JA/JW Tape 3592 E06 JJ/JR Tape	3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R
3592 E06 JB/JX Tape	3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R/W
3592 E07/EH7 JC/JY Tape 3592 E07/EH7 JK Tape	3592 E07/EH7 (TS1140) 3592 E08/EH8 (TS1150) 3592 55E/F/G (TS1155) 3592 60E/F/G (TS1160)	R/W R/W R R
3592 E07/EH7 JB/JX Tape	3592 E07/EH7 (TS1140)	R/W
3592 E08/EH8 JC/JY Tape 3592 E08/EH8 JD/JZ Tape 3592 E08/EH8 JK Tape 3592 E08/EH8 JL Tape	3592 E08/EH8 (TS1150) 3592 55E/F/G (TS1155) 3592 60E/F/G (TS1160)	R/W R/W R/W
3592 55E/F/G JD/JZ Tape 3592 55E/F/G JL Tape	3592 55E/F/G (TS1155) 3592 60E/F/G (TS1160)	R/W R/W
3592 60E/F/G JE/JV Tape 3592 60E/F/G JM Tape	3592 60E/F/G (TS1160)	R/W
3592 70F/S JF Tape	3592 70F/S (TS1170)	R/W
STK 9840A STK 9840B	STK 9840A STK 9840B STK 9840C STK 9840D	R/W R/W R R
STK 9840C	STK 9840C STK 9840D	R/W R
STK 9840D	STK 9840D	R/W
STK 9940A	STK 9940A STK 9940B	R/W R

Cartridge type	Drive preference list	Operation supported
STK 9940B	STK 9940B	R/W
STK T10000A	STK T10000A STK T10000B STK T10000C STK T10000D	R/W R R R
STK T10000B	STK T10000B STK T10000C STK T10000D	R/W R R
STK T10000C	STK T10000C STK T10000D	R/W R
STK T10000D	STK T10000D	R/W

Note: HPSS generally refers to Oracle StorageTek media and hardware as STK. The STK tape cartridges listed above refer to the HPSS imported media type. So, an "STK T10000A cartridge" is a cartridge that was imported into HPSS as an "STK T10000A" cartridge and thus its label was written by an STK T10000A drive.

Additionally, for STK T10000 cartridges, Oracle supports different recording format density cartridges: T1 and T2. HPSS does not distinguish between these different cartridge density types. It is up to the administrator to import the cartridge using the correct and allowable media type for the cartridge. The following table is provided to help the administrator understand where these types can be used. *Consult the Oracle StorageTek documentation for the most up-to-date information.*

StorageTek T10000 tape drive	T10000 cartridge (T1)	T10000 T2 cartridge (T2)
T10000A	Read/Write	N/A
T10000B	Read/Write	N/A
T10000C	Read Only	Read/Write
T10000D	Read Only	Read/Write

6.4.5. Disk devices

HPSS supports locally-attached disk devices, including those devices attached via SCSI, SSA, or Fibre Channel. For these devices, operating system disk partitions of the desired size must be created (for example, a Linux disk partition), and the raw device name must be used when creating the Mover device configuration (see the Configure a new device and drive section for details on configuring storage devices, including options that can improve performance).

In addition, HPSS supports writing to sparse files residing on an underlying file system. This allows for the use of storage technologies that support file system interfaces, such as optical disks, or other types of disk appliances.

6.4.6. AWS Tape Gateway

The AWS Tape Gateway is one of several storage gateways provided by AWS. The AWS Tape Gateway provides cloud-based virtual tape storage for on-premises applications. This solution allows data to be stored in S3 in either Glacier Flexible Retrieval or Glacier Deep Archive tiers. HPSS can be used with the AWS Tape Gateway as a way to move HPSS data into cloud storage.

The AWS Tape Gateway provides an emulated tape library and tape drives. It is important to note that AWS Tape Gateway does not support tape zoning, logical block protection, or SCSI reservations, but otherwise can be configured just like a real tape library using the SCSI PVR. See the AWS Tape Gateway section for details on configuring an AWS Tape Gateway tape storage class or tape device.

Note that the data stored via AWS Tape Gateway will not be visible in AWS as files. The end result is a set of tape images that can be restored by HPSS.

6.4.7. Special bid considerations

The following options are available by special bid only:

- HPSS High Availability

6.5. HPSS sizing considerations

There are two types of storage space that must be planned for: HPSS user storage space and HPSS infrastructure storage space.

HPSS user storage space is the disk and tape storage resources that will store user data. Typically, disk storage is allocated at the top level of storage hierarchies and is used as a disk cache for user data. Tape storage is usually allocated to lower levels of storage hierarchies and is used for the long-term, permanent storage of user data.

HPSS infrastructure storage space is the disk space allocated to file systems that contain executables, log files, server metadata (DB2 database), backups, and other HPSS supporting files and data. Tape resources outside of HPSS are usually required for backups of the operating system, HPSS specific file systems, and HPSS metadata unless other backup processes have been configured.

During the HPSS planning phase, it is important to assess how much disk space will be required to support the HPSS production environment. The first step in this process is to understand the various metadata tables managed by the HPSS system. The sections that follow explain the metadata table layout and how to best estimate disk space allocation to support these tables.

How these resources are interconnected to the overall system is just as important as the amount of disk or number of tape drives and cartridges allocated. For instance, if there are terabytes of disk storage in the form of several FC disk arrays and 50 enterprise type tape drives, but only one Mover and a couple of FC adapters, it is unlikely that the storage system will be able to adequately move data into, out of, and within the system to meet anyone’s demands and expectations. The "data pipes" between the storage resources must be adequately provisioned to allow for efficient transfer of data, including those times of peak demand. At the other extreme, one or both of the storage ends can be underallocated and waste the overall potential of the infrastructure. If there are too

few tape drives, data stalls on the disk resources preventing new files from being transferred into the storage system, or from being staged back from tape media in a timely manner when the user requests access to it.

HPSS is a relational database application directly dependent on the speed and performance characteristics of the storage assigned to it. The following considerations must be given to this storage:

- Sufficient operational space to handle the expected amounts of metadata and user data (stored separately).
- Sufficient I/O capacity to handle the expected workload.

High-performance disk and tape systems for user data storage must be accompanied by high-performance storage supporting the HPSS database operations.

HPSS has the capability to take advantage of Storage Area Networks. Though separated in HPSS generic configuration in reality there is usually only one SAN at an installation, and all the resources are attached to it. Besides the HPSS Movers being connected to SAN, the end-user clients are often SAN-attached as well. The result is that the data paths take greater advantage of the SAN architecture, fewer store-and-forward operations are needed through Movers (that is, clients transfer data across SAN directly to disk resources, the Mover just manages the transfer), and less traffic across the TCP/IP network infrastructure. Adequately provisioning the "data pipes" is still critical, but the equation has changed to rely more heavily on the SAN to carry the data traffic.

6.5.1. HPSS user storage space

HPSS files are stored on the media that is defined and allocated to HPSS. Enough storage space must be provided to meet the demands of the user environment. HPSS assists in the management of space by providing SSM screens with information about total space and used space in all of the defined storage classes. In addition, alarms can be generated automatically based on configurable threshold values to indicate when space used in a given storage class has reached a threshold level. In a hierarchy where data is being migrated from one hierarchy level to a lower one, management of space in the storage class provided is done via the migration and purge policies that are provided. The basic factors involved are the total amount of media space available in the storage class being migrated and the rate at which this space is used. This will drive how the migration and purge policies are set up for the storage class. For more details on this, see Migration policy and Purge policy.

Failure to provide enough storage space to satisfy a user request results in the user receiving a NO SPACE error. It is important to understand that the Core Server writes files only to the top level of the COS hierarchy. If the top level does not have sufficient free space for the write operation, it will fail, regardless of the amount of free space in lower levels.

6.5.2. HPSS infrastructure storage space

HPSS Core Server and metadata resources depicts a typical Core Server configuration and the metadata resources used by HPSS. The interconnect between the online storage resources and Core Servers varies on the specifics of a site, but usually falls into three categories: FC SAN, SAS, or Direct Connect. In all configurations, there is a certain amount of redundancy built into this configuration

with the use of Dual Controllers in each storage array, multiple connection paths from the Core Servers, and in the case of an FC SAN - multiple SAN switches would be recommended. Additionally, key DB2 components like the DB2 log and DB2 logmirror would be separated onto different storage units. The goal of the configuration is to provide the greatest amount of protection and availability by using redundant components and strategically separating key resources across the hardware components. This is extended further with the use of a spare machine (for Manual Failover) or HADR system to provide redundancy for HPSS services as well.

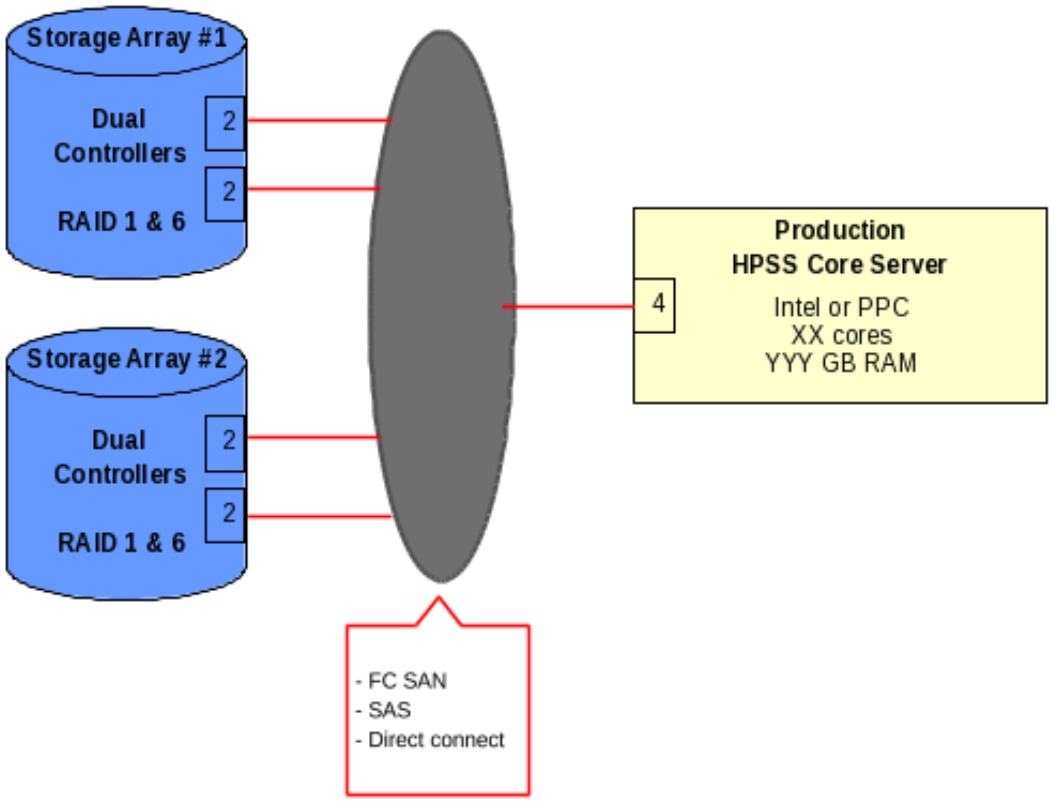


Figure 5. HPSS Core Server and metadata resources

Metadata disk layout - Rack 1 and Metadata disk layout - Rack 2 show a possible allocation scheme using two E5624 with two trays of disks for each disk array.

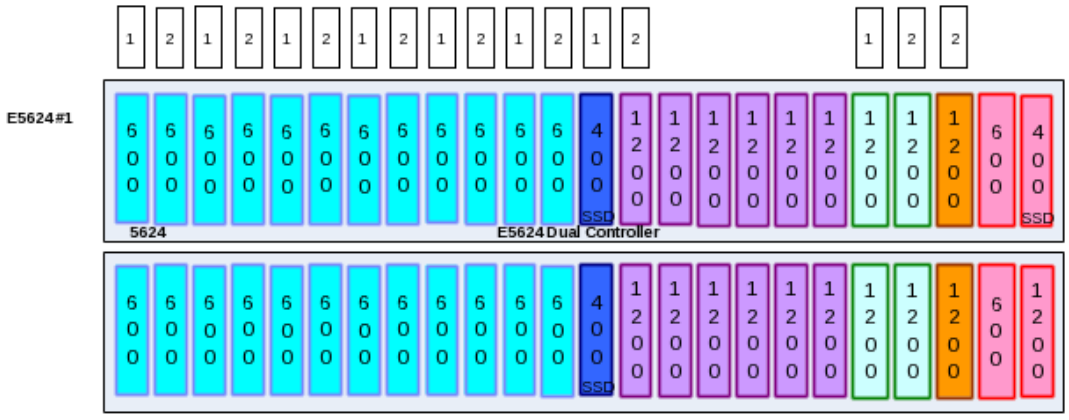


Figure 6. Metadata disk layout - Rack 1

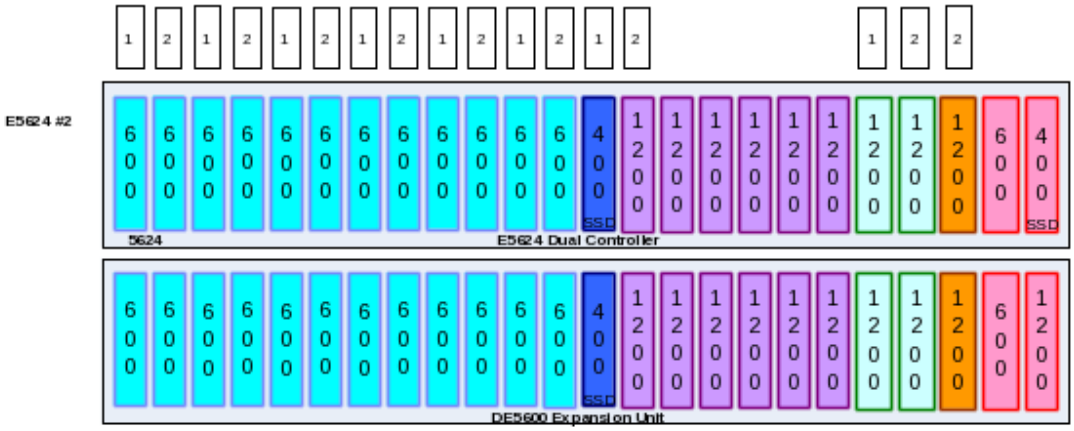


Figure 7. Metadata disk layout - Rack 2

Disk Array #1 configuration uses the following grouping of the disk resources:

- Twelve (1+1) RAID-1 arrays for the odd DB2 storage paths (light blue)
- One (1+1) RAID-1 array for the first copy of DB2 log (dark blue)
- One (10+2) RAID-6 for half of the DB2 backup space (purple)
- Two (1+1) RAID-1 arrays for HPSS file systems (green)
- One (1+1) RAID-1 array for DB2 log archive Mirror (brown)
- Four Hot Spare disks (red) (size and type vary)

Disk Array #2 configuration uses the following grouping of the disk resources:

- Twelve (1+1) RAID-1 arrays for the even DB2 storage paths (light blue)
- One (1+1) RAID-1 array for the second copy of DB2 log MIRROR (dark blue)
- One (10+2) RAID-6 for half of the DB2 backup space (purple)
- Two (1+1) RAID-1 arrays for HPSS file systems (green)
- One (1+1) RAID-1 array for DB2 log archive (brown)
- Four Hot Spare disks (red) (size and type vary)



The array model, disk sizes and types (spinning disk versus SSD), and grouping will vary with sites depending upon their operational requirements and available HW components. Specifics are developed with the HPSS Systems Engineering team and HPSS DB2 SMEs as needed. The above detail and tables shown in HPSS file systems are to only provide an example of how two storage arrays could be configured for a system that is designed to manage up to 500 million files. There are many considerations that must be taken into account to define the metadata storage which are well beyond the scope of this document. The examples, therefore, should only be used to understand the concept, and not used as an exact template for your site's situation.

The majority of the resources (light blue) will be allocated for DB2 metadata storage — this is

storage for the HSUBSYS1 database. (The example does not cover the use of multiple subsystems nor the use of database partitioning.) Data is protected using a 1+1 RAID-1 configuration. Multiple LUNs are used, and reasonably high I/O performance can be attained across these resources; higher than what could be achieved if using multiple disks in a RAID 5 or RAID 6 configuration.

The DB2 log and logmirror also utilize a 1+1 RAID-1 configuration. While the RAID-1 provides data protection against a single disk failure, the log and logmirror are split between the two disk array units to provide additional protection against the loss of the entire disk array. This same arrangement is also in place for the first and second copy of the log-archives. This configuration redundancy eliminates a "single point of failure" in DB2 recovery objects (database backup images should have multiple copies as well) and allows the DB2 metadata databases to be recovered to the point of failure in the event of a complete disk array unit loss.

The remaining resources are allocated to the DB2 backup file system and HPSS supporting file systems (that is, where HPSS is installed). For the DB2 backup, the goal is to provide the maximum amount of space for the given set of resources, so a RAID-6 is used. Performance for this file system needs to be "good", but available space is more important. The HPSS supporting file systems have both lower space and performance requirements, but do need to be separate from other DB2 uses.

6.5.2.1. HPSS and DB2 file systems

The following subsections describe the various file systems used by HPSS.

/opt/hpss

The HPSS software is installed directly in the `/opt/hpss` directory. HPSS version information can be determined using the `rpm -q hpss` command or by checking the HPSS version file.

/var/hpss

The /var/hpss files section contains detailed explanation of directories and files located in `/var/hpss`.

The `/var/hpss` directory tree is the default location of a number of HPSS configuration files and other files needed by the servers. It is recommended that this file system be at least 64 GB in size. A symbolic link `/var/hpss` references the `/varhpss` file system. Within the `/var/hpss` file system the following subdirectories exist:

- The `/var/hpss/etc` is the default directory where some additional UNIX configuration files are placed. These files are typically very small.
- The `/var/hpss/ftp` is the default directory where several PFTP daemon files are maintained. There are three subdirectories required: `adm` (contains the `hpss_ftpd.log` file), `daemon`, and `daemon/ftpd` where the `ftp.pids-hpss_class` file will be placed.
- The `/var/hpss/tmp` is the default directory where the Startup Daemon creates a lock file for each of the HPSS servers started on the node. HPSS may also write diagnostic log files and disk allocation maps in this directory, when configured to do so. The lock files are very small, but the logs and disk maps may be several tens of kilobytes, or larger.

  It is up to the administrator to remove unneeded reports to prevent the `/var/hpss` file system from filling.

- The `/var/hpss/log` is the directory for HPSS log files. It is the recommended location for HPSS-specific log files. The `/var/hpss/log/HPSS.log` file will be created here when you run the `configure_logging` tool during installation (see Configure Logging), unless logging is being redirected elsewhere in SSM. Additional HPSS log files may also appear here.
- If the MPS is configured to produce a migration/purge report, it will generate a report every 24 hours. The size of these reports varies depending on the number of files being migrated and purged during the report cycle. These reports are placed in the `/var/hpss/mps` directory by default and will need to be removed when no longer needed.

  It is up to the administrator to remove unneeded reports to prevent the `/var/hpss` file system from filling.

- If the Gatekeeper is configured to do Gatekeeping Services, then the administrator may wish to create a site policy configuration file, usually named `/var/hpss/gk/gksitepolicy`. The size of this file depends on the site-implemented gatekeeping policy. If the Gatekeeper Service is not used, there is a minimal amount of disk space used in this directory.
- If an accounting report is requested, a report file and a checkpoint file are created in the directory specified in the accounting policy, usually `/var/hpss/acct`. The sizes of these files depend upon the number of files stored in HPSS.

  It is up to the administrator to remove unneeded reports to prevent the `/var/hpss` file system from filling.

- If SSM is configured to buffer alarm and event messages in a disk file, a file to store alarms and events will be created in `/var/hpss/ssm` directory. This alarm file is approximately 5 MB in size.

/hpss_corefile

is the default directory where HPSS servers put "core" files if they terminate abnormally. A symbolic link `/var/hpss/adm/core` is created to point to `/hpss_corefile`. Core files may be large, so it is recommended that there should be at least 64 GB reserved for this purpose on the server node and at least 4 GB on Mover nodes.

It is up to the administrator to remove unneeded core files to prevent the `/hpss_corefile` file system from filling.

/opt/ibm/db2

This is deployed as a symbolic link to a file system called `/optdb2`.

`/opt/ibm/db2` is the file system where the DB2 product is installed. Different versions of DB2 can be installed at the same time, and the size should be at least 64 GB.

The following DB2 file systems have a common base directory `/db2data`. This is a change from earlier releases of HPSS, but in an effort to help prevent accidental removal of critical DB2 files, we want the path to clearly identify this as DB2 data and users to be extra vigilant when operating in this part of the name space.

/db2data/db2_hpssdb

The `/db2data/db2_hpssdb` file system stores the DB2 instance configuration information and CFG database tables. It is also the HPSS instance home directory. Specifics on its size are described in the *CFG Database Allocation* section below. We recommend a file system size of 64 GB.

/db2data/p{0000, ..., p-1}/stg{0001, ..., t}

The DB2 storage path file systems are where DB2 stores the HPSS metadata. It takes the above form where "p" is the number of partitions, and "t" is the number of storage path file systems.

For most sites, the number of partitions will be 1, and there will be only one subsystem. The number of storage paths will be determined during the planning phase and the specifics of the metadata HW used for the site. The `/db2data/p#\##\#/stg#\###` file systems store the bulk of the HPSS metadata which is used to define the HPSS name space, bitfile information, disk and tape segments, disk and tape volumes, and activity information for migration/purge operations. When defining a new system, the HPSS Systems Engineering team needs to be consulted to help define or verify a configuration that will meet both the capacity and transaction requirements of the storage system.

Sites that have, or will have, hundreds of millions or billions of files, should consider the use of the DB2 Database Partitioning Feature (DPF), subsystems, or both. Planning for these features is beyond the scope of this document and will require additional consultation with the HPSS Systems Engineering and HPSS DB2 SME teams to customize a suitable configuration specific to the site's unique requirements.

/db2data/p{0000, ..., p-1}/db2_backup1 & db2_backup2

The `/db2data/p#\##\#/db2_backup1` and `/db2data/p#\###/db2_backup2` file systems temporarily store backup images of the CFG and SUBSYS{X} databases. Typically subdirectories are used to segregate online versus offline backups, as well as separate subdirectories for each database. Something similar to the following:

```
⋯/online/cfg
⋯/online/subsys1
⋯/offline/cfg
⋯/offline/subsys1
```

Two file systems are used to store two complete backups on separate storage resources to protect the images from a single file system failure. After the backup images are generated, they are then transferred to long-term media, such as tape, using a backup file manager such as TSM. Details are described in HPSS metadata space.

Typically, backups are generated nightly, and at least a week's worth of backups should be kept on local file systems. Refer to the *IBM HPSS Operational Disaster Prevention Plan* document and consult with the HPSS Systems Engineering team to ensure enough space is reserved for this file system to support proper backups.

/db2data/p{0000, ..., p-1}/db2_log & db2_logmirror

These two file systems contain the active transaction logs for the HPSS databases. The underlying storage for these file systems should have similar performance characteristics since DB2 transactions will wait for both log entries to be written. A separate subdirectory is defined under each file system for the CFG database and the SUBSYS databases:

```
⋯/cfg
⋯/subsys1
```

These file systems are typically backed by resources that support high-transaction I/O rates (for example, solid-state devices (SSD)). The amount of space required is usually small, a few tens of gigabytes, but I/O performance is critical to support HPSS metadata operations. For those sites with multiple subsystems, additional file systems (with separate storage resources) may be required for optimal performance. The "subdirectories" for the subsystems (subsys[2-5]) would then become mount points for these additional file systems.

Also, it is critical for the two file systems (`/db2data/db2_log` and `/db2data/db2_logmirror`) to be on separate metadata storage resources. Not only for performance but also for data protection. Loss of both file systems puts HPSS metadata at severe risk. Therefore, the "log" and "logmirror" are not only put on separate LUNs, but are required to be on separate storage arrays with dual paths to the controllers on each array to provide complete separation in the event of a catastrophic loss of a complete storage array.

/db2data/p{0000, ..., p-1}/db2_logarchive1 & db2_logarchive2

Similar to DB2 log and logmirror file systems, two separate file systems are required to store completed ("archived") DB2 logs as they are generated. These files, in conjunction with the DB2 online backups, provide DB2 with the ability to recover DB2 to the point of failure in the event the primary DB2 resources are damaged, destroyed, or otherwise unavailable. The files in both file systems need to be safely copied to other media/storage like the DB2 backup images. The amount of storage for each file system should be adequate to store at least one week's worth of log archives, but more is recommended. It is up to the administrator to maintain the space in these file systems by periodically removing log archives that:

1. have been safely copied elsewhere, or
2. are no longer referenced by DB2 backup history.

Refer to the *IBM HPSS Operational Disaster Prevention Plan* document for more information regarding the backup and restore process.

6.5.2.2. HPSS metadata space

During the HPSS planning phase, it is important to properly assess how much storage space and I/O capacity will be required by DB2 to manage HPSS metadata. The first step in this process is to understand the storage requirements for DB2. The sections that follow explain the various DB2 uses for storage and how best to estimate disk space allocation and capability to support DB2/HPSS operations.

CFG database allocation

By default, **mkhpss** will store the DB2-related files for HPSS in the `/db2data/db2_hpssdb` file system. As recommended above, this directory should be a separate file system of RAID disks. The amount of space needed will vary somewhat depending upon the number of subsystems in use. For a site with only one subsystem, the amount of space should be at least 64 GB.

SUBSYS database allocation

As a general rule of thumb, approximately 3500 bytes per file of storage should be reserved for the HPSS metadata storage paths. This general rule of thumb assumes:

- Table and index compression is enabled on large tables.
- Hierarchies with disk plus two levels of tape.
- A disk cache of no more than 10% of the total storage of the system.
  - With a reasonable/low number (less than 10 segments) of disk storage segments defined per file.
  - Systems that have a large number of disk segments or files with many holes may consume more storage space per file.
  - Systems that utilize UDAs may consume more storage space.

The calculations include additional space to allow for future in-place conversions and updates, since those may (or may not) utilize the same resources.

For sites with hundreds of millions or billions of files, the required disk space will require additional analysis since database partitioning will likely need to be utilized. The number of partitions will need to be discussed early in the planning stages of a new system, or a major upgrade. Consultation with the HPSS Systems Engineering and DB2 SME teams is required. Background information on database partitioning can be obtained from HPSS support, or it can be found by reviewing the DB2 information presented at the annual HPSS Users Forums, which are linked to from the HPSS Administrator Wiki.

DB2 log / DB2 log mirror

Recommended minimum: 200 GB for each file system per subsystem. Normally 10 log files (100 MB each) are required for each subsystem, but additional logs may be kept in the following situations:

- periods of heavy workloads
- long-running transactions
- when the log archive path is unavailable

DB2 log archives

Recommended minimum: 500 GB for each file system per subsystem, but this is dependent upon how busy the system is. Busier systems will need to have more space available. There should be at least enough space to keep one week's worth of log archive files (uncompressed) available for recovery. The HPSS Systems Engineering team will provide more guidance on appropriate sizing.

DB2 backup

Recommended minimum: 4 TB per subsystem for DB2 backup images that will be stored online; large subsystems will require additional space. Keep in mind this file system will contain separate nightly backup images for the last seven days (that is, a week; more is better).

Other file systems

Typically, the remaining file systems (`/var/hpss`, `/hpss`, `/opdb2`, `/hpss_corefile`, `/db2data/db2_hpssdb`) can be created out of shared LUN. There is no driving need for large amounts of space, nor high performance, for these file systems.

6.5.2.3. HPSS file systems

The following table provides a summary of what the file systems would look like (using the example of the E5624 disk arrays found earlier in this section).

Table 4. HPSS and DB2 file systems

HPSS and DB2 file systems There is one DB2 server				
LV name or device	**VG**	**Size ***	**File system mount point**	**Symlink to file system**
varhpss	hpss_vg	64 GB	/varhpss	/var/hpss
opt_hpss	hpss_vg	64 GB	/opt/hpss	
optdb2	hpss_vg	64 GB	/optdb2	/opt/ibm/db2
hpss_corefile	hpss_vg	64 GB	/hpss_corefile	/var/hpss/adm/core
db2_hpssdb	hpss_vg	64 GB	/db2data/db2_hpssdb	
db2_backup1	db2_backup1_vg	10.7 TB	/db2data/db2_backup1	
db2_backup2	db2_backup2_vg	10.7 TB	/db2data/db2_backup2	
db2_log	db2_log_vg	361 GB	/db2data/db2_log	
db2_logmirror	db2_logmirror_vg	361 GB	/db2data/db2_logmirror	
db2_logarch1	db2_logarch1_vg	1094 GB	/db2data/db2_logarchive1	
db2_logarch2	db2_logarch2_vg	1094 GB	/db2data/db2_logarchive2	
db2_stg0001	db2_stg0001_vg	545 GB	/db2data/stg0001	
db2_stg0002	db2_stg0002_vg	545 GB	/db2data/stg0002	

HPSS and DB2 file systems There is one DB2 server				
. . .	. . .	. . .	. . .	
db2_stg0024	db2_stg0024_vg	545 GB	/db2data/stg0024	

Mapping the LVs to LUN labels is shown in the following table:

Table 5. LV To LUN label mapping

LUN label	VG	LUN label	VG
db2_stg0001	db2_stg0001_vg	db2_stg0002	db2_stg0002_vg
db2_stg0003	db2_stg0003_vg	db2_stg0004	db2_stg0004_vg
. . .	. . .	. . .	. . .
db2_stg0023	db2_stg0023_vg	db2_stg0024	db2_stg0024_vg
db2_log	db2_log_vg	db2_logmirror	db2_logmirror_vg
db2_backup1	db2_backup1_vg	db2_backup2	db2_backup2_vg
hpssfs1-1	hpss_vg	hpssfs2-1	hpss_vg
hpssfs1-2	hpss_vg	hpssfs2-2	hpss_vg
db2_logarch1	db2_logarch1_vg	db2_logarch2	db2_logarch2_vg

And LUN labels map back to the storage arrays given the following two tables:

Table 6. Storage Array #1

Array	Vol ID	Volume label	~Size	Config	Purpose
db2_stg0001	0	db2_stg0001	594 GB	1+1 RAID1	DB2 Storage
db2_stg0003	1	db2_stg0003	594 GB	1+1 RAID1	DB2 Storage
. . .	. . .	. . .	. . .	. . .	. . .
db2_stg0023	11	db2_stg0023	594 GB	1+1 RAID1	DB2 Storage
db2_log	12	db2_log	394 GB	1+1 RAID1	DB2 Log
db2_backup1	13	db2_backup1	11943 GB	10+2 RAID6	DB2 Backup
hpssfs1-1	14	hpssfs1-1	1194 GB	1+1 RAID1	HPSS File systems
hpssfs1-2	15	hpssfs1-2	1194 GB	1+1 RAID1	HPSS File systems
db2_logarch1	16	db2_logarch1	1194 GB	1+1 RAID1	DB2 Log Archive

Table 7. Storage Array #2

Array	Vol ID	Volume label	~Size	Config	Purpose
db2_stg0002	0	db2_stg0002	594 GB	1+1 RAID1	DB2 Storage
db2_stg0004	1	db2_stg0004	594 GB	1+1 RAID1	DB2 Storage
. . .	. . .	. . .	. . .	. . .	. . .
db2_stg0024	11	db2_stg0024	594 GB	1+1 RAID1	DB2 Storage
db2_log	12	db2_log	394 GB	1+1 RAID1	DB2 Log
db2_backup2	13	db2_backup2	11943 GB	10+2 RAID6	DB2 Backup
hpssfs2-1	14	hpssfs2-1	1194 GB	1+1 RAID1	HPSS File systems
hpssfs2-2	15	hpssfs2-2	1194 GB	1+1 RAID1	HPSS File systems
db2_logarch2	16	db2_logarch2	1117 GB	1+1 RAID1	DB2 Log Archive

6.5.3. System memory and disk space

The following sections discuss recommendations and requirements for disk space, system memory, and paging space.

6.5.3.1. Operating system disk spaces

It is recommended that all operating system logical volumes and partitions be mirrored. This is true of the HPSS server and Mover nodes.

6.5.3.2. System disk space requirements for running SSM

The SSM Graphical User Interface, hpssgui(7), and Command Line Interface, hpssadm(7), have an option to create session log files. hpssgui(7) records all status bar and pop-up messages issued during the session in its log. hpssadm(7) records all prompts, error and informational messages, and requested output (for example, lists, managed objects, and configuration structures) issued during the session in its log. Old session log files should be removed periodically to avoid filling the file system. This is typically done with a **cron** job. For example, the following command will remove all files from `/tmp` which have not been accessed within the previous seven days:

```
% find /tmp -atime +7 -exec rm {} \;
```

The creation of the session log files is controlled by the **-S** option to the hpssgui(7) and hpssadm(7) startup scripts.

The SSM API will log access and error messages in /var/hpss/log. These logs do not auto-rotate and require configuration with a log rotation policy, which is discussed in detail in Recommendations for syslog configuration.

6.5.3.3. System memory and paging space requirements

The memory and disk space requirements for the nodes where the HPSS servers will execute depends on the configuration of the servers, the nodes that each server will run on, and the amount of concurrent access they are configured to handle.

At least 2 GB of memory is recommended for nodes that will run one or more HPSS servers (and most likely a DB2 server), excluding the HPSS Movers. More memory is required for systems that run most of the servers on one node or support many concurrent users (or both). The memory available to HPSS and DB2 servers is critical to providing acceptable response times to end-user operations. Disk space requirements are primarily covered by HPSS metadata space for DB2 space, and the preceding subsections under System memory and disk space for the individual HPSS servers. Sufficient disk space should be allocated for the paging space, using recommendations in the system documentation for the amount of memory configured.

The amount of memory for nodes running HPSS Movers, and no DB2 servers, is dependent on the number and types of devices configured on the Mover node, the expected usages of those devices, and the configuration of the Movers. In general, Movers supporting disk devices will require more memory than Movers supporting tape devices because disk devices are likely to have more outstanding requests. At least 1 GB of memory should be configured on the Mover nodes. More memory is required for nodes that support many devices, especially disks, and have large numbers of concurrent end-user requests. Additionally, the size of the Mover's internal buffers impacts the Mover's memory requirements. Two buffers are used by each Mover process to handle I/O requests.

Paging space should be sized according to the following rules:

Table 8. Paging space info

Amount of physical memory	Minimum recommended amount of paging space
memory <= 256 MB	2.0 × amount of physical memory
256 MB < memory <= 1 GB	512 MB + ((amount of physical memory : 256 MB) × 1.25)
1 GB < memory <= 2 GB	1.5 × amount of physical memory
memory > 2 GB	1.0 × amount of physical memory

6.6. HPSS interface considerations

This section describes the user interfaces to HPSS and the various considerations that may impact the use and operation of HPSS.

6.6.1. Client API

The HPSS Client API provides a set of routines that allow clients to access the functions offered by HPSS. The API consists of a set of calls that are comparable to the file input/output interfaces defined by the POSIX standard (specifically *ISO/IEC 9945-1:1990* or *IEEE Standard 1003.1-1990*), as well as extensions provided to allow access to the extended capabilities offered by HPSS.

The Client API is built on top of the HPSS security layer (either UNIX or Kerberos). It must be run on a platform that supports the Core Server's security layer. For example, if the Core Server is using Kerberos authentication then users on the client platform must be able to authenticate themselves with the Core Server's Kerberos realm. To access HPSS from client platforms that do not support the Core Server's security layer, FTP or Parallel FTP must be used.

The Client API allows clients to specify the amount of data to be transferred with each request. The amount requested can have a considerable impact on system performance and the amount of metadata generated when writing directly to a tape storage class. See HPSS performance considerations for further information.

The details of the Application Program Interface are described in the *HPSS Programmer's Reference*.

6.6.2. FTP

HPSS provides an FTP daemon that supports standard FTP clients. Extensions are also provided to allow additional features of HPSS to be utilized and queried. Extensions are provided for specifying Class of Service to be used for newly created files, as well as directory listing options to display Class of Service and accounting code information. In addition, the **chgrp**, **chmod**, and **chown** commands are supported as **quote site** options.

The FTP daemon is built on top of the Client API and must be run on a node that supports Kerberos clients. Note that FTP clients can run on computers that do not have Kerberos installed.

The size of the buffer used for reading and writing HPSS files can be specified in the FTP daemon configuration. The buffer size selected can have a considerable impact on both system performance and the amount of metadata generated when writing directly to a tape storage class. See HPSS performance considerations for further information.

The GSSFTP from MIT is supported if the HPSS FTP daemon is appropriately configured. This client provides credential-based authentication and "Cross Realm" authentication to enhance security and "password-less" FTP features.

Refer to the FTP/PFTP daemon configuration section for details on configuring the FTP daemon.

Refer to the *HPSS User's Guide* for details of the FTP interface.

6.6.3. Parallel FTP

The FTP daemon also supports the HPSS Parallel FTP (PFTP) protocol, which allows the PFTP client to utilize the HPSS parallel data transfer mechanisms. This provides the capability for the client to transfer data directly to the HPSS Movers (that is, bypassing the FTP daemon), as well as the capability to stripe data across multiple client data ports (and potentially client nodes). Data transfers are supported through TCP/IP. Support is also provided for performing partial file transfers.

The FTP protocol is supported by the HPSS FTP daemon. Refer to the FTP/PFTP daemon configuration section for configuration information. No additional configuration of the FTP daemon is required to support PFTP clients.

The client-side executable for PFTP is pftp_client(7), which supports TCP-based transfers. Because the client executable is a superset of standard FTP, standard FTP requests can be issued as well as the PFTP extensions. Authentication using either username/password or Kerberos credentials is configurable.

Refer to the *HPSS User's Guide* for details of the PFTP interface.

6.7. HPSS server considerations

Servers are the internal components of HPSS that provide the system's functionality. Each HPSS server executes as one or more UNIX processes. They must be configured correctly to ensure that HPSS operates properly. This section outlines key considerations that should be kept in mind when planning the server configuration for an HPSS system.

6.7.1. Core Server

The Core Server is responsible for managing the HPSS name space (such as files, directories, and links), bitfiles, and storage (such as physical volumes and virtual volumes) for a single subsystem. Each of these areas of responsibility are outlined in greater detail below.

Core Server at large

The Core Server uses POSIX threads to service concurrent requests. The Core Server accepts requests from any authenticated client; however, certain Core Server functions can be performed only by trusted clients. Trusted clients are those for whom control permission has been set in the Core Server's ACL entry for the client. Higher levels of trust are granted to clients who have both control and write permission set in their ACL entry. Refer to the HPSS server security ACLs section for information concerning the ACL for the Core Server.

The Core Server can be configured to allow or disallow super-user privileges (root access). When the Core Server is configured to allow root access, the UID of the super-user is configurable.

HPSS systems configured with multiple subsystems employ multiple Core Servers and multiple metadata databases. Though the servers are separate, each Core Server in a given HPSS realm must share the fileset global metadata table.

Name space

The HPSS Core Server maintains the HPSS name space in system metadata. Refer to HPSS sizing considerations for details on sizing the name space. Refer to the DB2 space shortage section for information on handling a metadata space shortage. By utilizing multiple storage subsystems, it is possible to distribute large name spaces across multiple Core Servers. Junctions between the names spaces of these storage subsystems can be used to "join" these subsystems.

Bitfiles

The Core Server provides a view of HPSS as a collection of files. It provides access to these files and maps the files onto underlying storage objects.

When a Core Server is configured, it is assigned a server ID. This value should never be changed because it is embedded in the ID of each bitfile and storage segments it uses. HPSS expects to be able to extract the server ID from any bitfile ID and connect to that server to access the file.

The Core Server maps bitfiles to their underlying physical storage by maintaining information that maps a bitfile to the storage segments that contain its data. For additional information, see Core Server. The relationship of bitfiles to storage segments and other structures is shown in The relationship of various server data structures.

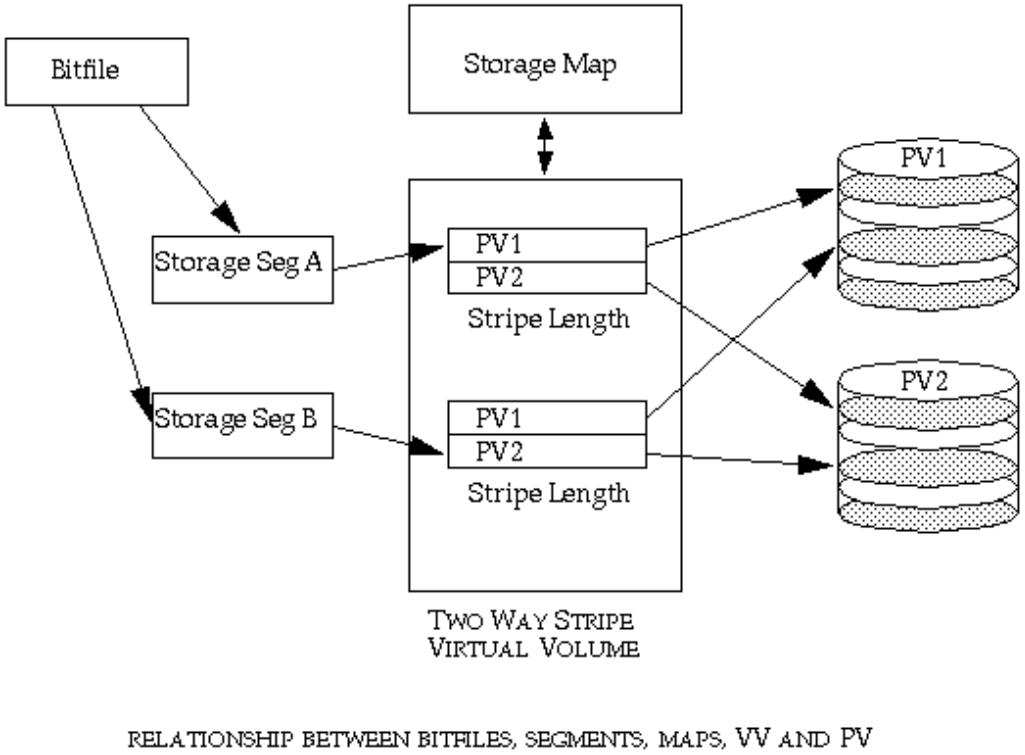


Figure 8. The relationship of various server data structures

Disk storage allocation

Each Core Server manages disk storage units for HPSS. It maps each disk storage unit onto an HPSS disk Virtual Volume (VV) and records configuration data for the VV. The server also maintains a storage map for each VV that describes which portions of the VV are in use and which are free. The relationship of various server data structures shows the relationship of Core Server data structures such as VVs to other server data structures.

When one disk unit is mapped to a VV, the disk is said to be in a "1-wide stripe". Files are written to the disk in the conventional fashion. When two or more disks are grouped together in a VV, the disks are said to be in an "n-wide stripe". Files written to the VV are striped across the disks. All disks are written simultaneously with different portions of the files. Striping data this way increases throughput to the disks by almost a factor of N, where N is the width of the stripe.

The server can manage information for any number of disk VVs; however, because a copy of all of the VV and storage map information is kept in memory while the server runs, the size of the server will be somewhat proportional to the number of disks it manages.

The Core Server is designed to scale up its ability to manage disks as the number of disks increase.

As long as sufficient memory and CPU capacity exist, threads can be added to the server to increase its throughput. Additional subsystems can also be added to a system, increasing concurrency even further.

See the section on disk I/O operations metrics for more information about monitoring how I/O is load-balanced across disk movers and VVs in a running HPSS system: HPSS Disk Mover and VV Load-Balancing

Tape storage allocation

Each Core Server manages magnetic tape storage media for HPSS. The server maps each tape cartridge onto an HPSS tape Physical Volume (PV) and records configuration data for the PV. Groups of one or more PVs are managed by the server as tape Virtual Volumes (VVs). The server maintains a storage map for each VV that describes how much of each tape VV has been written and which storage segment, if any, is currently writable at the end of the VV. The relationship of various server data structures shows the relationship of data structures such as VVs to other server data structures.

When one tape cartridge is mapped to a VV, the tape is said to be in a "1-wide stripe". Files are written to the tape serially in the conventional fashion. When two or more tapes are grouped together in a VV, the tapes are said to be in an "n-wide stripe". Files written to the VV are striped across the tapes. All tapes are written simultaneously with different portions of the files. Striping data this way increases throughput to the tapes by almost a factor of N, where N is the width of the stripe.

RAIT tape VVs are special cases of striped tape VVs. In a RAIT VV, one or more of the PVs is used to store parity information for each stripe of data that is written. This allows the system to recover data from the RAIT VV if some number of the individual tapes become unreadable. It should be understood that the parity information rotates around the tapes, stripe by stripe, in roughly RAID 5 or 6 fashion, so it is not strictly correct to think of one of the PVs as the "parity tape".

The server can manage information for any number of tape VVs. It can also manage an unlimited number of tape VVs, maps, and storage segments without impacting its memory size.

The Core Server is designed to scale up its ability to manage tapes as the number of tapes increases. As long as sufficient memory and CPU capacity exist, threads can be added to the server to increase its throughput. Additional subsystems can also be added to a system, increasing concurrency even further.

Note that the number of tape drives the server manages has much more to do with the throughput of the server than the number of tape volumes the server manages. If the number of tape drives in the system needs to increase to meet workload demands, adding a new subsystem and redistributing the drives may be the best way to deal with the increased workload.

Location Services

All HPSS client API applications, which includes all end user applications, will need to contact a Core Server at least once during initialization and usually later during execution in order to locate the appropriate servers to contact. If the Core Server is down for an extended length of time, these applications will eventually give up retrying their requests and become non-operational. Consider

increasing the automatic restart count for failed servers in SSM.

If any server is down for an extended length of time it is important to mark the server as non-executable within SSM. As long as a server is marked executable the Core Server continues to advertise its location to clients which may try to contact it.

6.7.2. Migration/Purge Server

The Migration/Purge Server (MPS) reports storage class usage statistics and manages the amount of free space available in storage classes by performing periodic migration and purge runs on the storage classes. Migration copies data from the storage class on which it runs to one or more lower levels in the storage hierarchy. Once data has been migrated, a subsequent purge run will delete the data from the migrated storage class, if so configured. Migration is a prerequisite for purge, and MPS will never purge data which has not previously been migrated. It is possible, but not desirable, to assign only a migration policy and no purge policy to a disk storage class; however, this will result in data being copied (migrated) but never deleted (purged). It is important to recognize that migration and purge policies determine when data is migrated from a storage class and when the data is purged from that storage class; however, the number of copies and the location of those copies is determined by the storage hierarchy definition. The MPS uses the Core Server both to perform data movement between hierarchy levels and gather statistics. As such, the Core Server must be running in order for the MPS to complete its duties.

The MPS can exist only within a storage subsystem and a subsystem may be configured with no more than one MPS. Storage hierarchies are global across all storage subsystems within an HPSS system, but a given hierarchy may or may not be enabled within a given subsystem (a hierarchy is enabled within a subsystem by configuring the subsystem to enable one or more classes of service which reference that hierarchy). Note that a storage class may not be selectively migrated and purged in different subsystems. If the hierarchy contains storage classes which require migration and purge, then an MPS must be configured to run against those storage classes in the subsystem. This MPS will manage migration and purge operations on only those storage resources within its assigned subsystem. Thus, for an HPSS system with multiple storage subsystems, there may be multiple MPSs, each operating on the resources within a particular subsystem.

Migration and purge operate differently on disk and tape storage classes. Disk migration and disk purge are configured on a disk storage class by associating a migration policy and a purge policy with that storage class. For tape storage classes, the migration and purge operations are combined and are collectively referred to as tape migration. Tape migration is enabled by associating a migration policy with a tape storage class. Purge policies are not needed or supported on tape storage classes.

Once migration and purge policies are configured for a storage class (and the MPS is restarted), the MPS will begin scheduling migration and purge runs for that storage class. Migration on both disk and tape is run periodically according to the runtime interval configured in the migration policy. Disk purge runs are not scheduled periodically, but rather are started when the percentage of space used in the storage class reaches the threshold configured in the purge policy for that storage class. It is critical that the hierarchies to which a storage class belongs be configured with proper migration targets in order for migration and purge to perform as expected.

The purpose of disk purge is to maintain a given amount of free space in a disk storage class by

removing data of which copies exist at lower levels in the hierarchy. The order in which purge records are sorted, which determines the order in which files are purged, may be configured on the purge policy. It should be noted that all of the options except *Record Create Time* require additional metadata updates and can impose extra overhead on DB2. Also, unpredictable purge behavior may be observed if the purge record ordering is changed with existing purge records in the system until these existing records are cleared. A purge run ends when either the supply of purge records is exhausted or the purge target is reached.

Tape file migration is a file-based tape method which is able to make a single copy of tape files to the immediately lower level in the hierarchy. For details on tape migration options, see Migration policy.

If the MPS detects that the source tape volume has become active at any point during a tape file migration run, migration is abandoned on this volume until the next migration run. This is done in order to avoid competing with an HPSS system user for this volume. A tape volume is deemed to be active if any file it contains has been read or written within the access intervals specified in the migration policy.

The MPS provides the capability of generating migration/purge report files that document the activities of the server. The specification of the UNIX report file name prefix in the MPS server-specific configuration enables the server to create these report files. It is suggested that a complete path be provided as part of this file name prefix. Once reporting is enabled, a new report file is started every 24 hours. The names of the report files are made up of the UNIX file name prefix from the server-specific configuration, plus a year-month-day suffix. With reporting enabled, MPS will generate file-level migration and purge report entries in real time. These report files can be interpreted and viewed using the mps_reporter(7) utility. Since the number and size of the report files grow rapidly, each site should develop a **cron** job that will periodically remove the reports that are no longer needed.

In order to efficiently perform disk migration, the MPS migrates files in parallel from disk to tape. The number of streams that the MPS uses simultaneously can be configured by the user using the Migration Stream Count in the *Disk Migration Policy* screen. For example, if the Migration Stream Count is set to one, then the MPS will serially migrate aggregates. If the Migration Stream Count is set to four, then the MPS will attempt to have four aggregates migrating at one time. The actual number of virtual volumes used depends on the total stripe width of the storage class configuration.

Tape aggregation, the default behavior for the disk migration policy, is the process of aggregating (that is, bundling together) many files on tape. This can dramatically increase migration performance, especially for sites that tend to ingest many relatively small files. When an individual file is written to tape, its data is flushed before the next file is written. When files are aggregated on tape, the time to flush the data to tape is only incurred once for the entire aggregate.

The Ordered Migration feature orders files being migrated from disk to tape by directory or by creation time. Tape aggregates can be used to group files according to their directories. This is especially pertinent when considering how to employ the Full Aggregate Recall feature. The objective of Ordered Migration is to co-locate data on tape, with either the directory of the files or the files' create times being the co-location criteria. A site may want to combine this feature with Full Aggregate Recall to potentially minimize the number of tape operations required to bring a

directory's files back from tape. Of course, the migration policy will continue to honor the migration policy settings for when a file is eligible for migration and how often migration will run. Refer to the Class of Service Configuration window and Migration policies sections for further details on this feature.

As previously indicated, the MPS provides the information displayed in the *HPSS Active Storage Classes* window in SSM. Each MPS contributes storage class usage information for the resources within its storage subsystem. MPS accomplishes this by polling the Core Server within its subsystem at the interval specified in the MPS server-specific configuration. The resulting output is one line for each storage class for each storage subsystem in which that class is enabled. The MPS for a subsystem does not report on storage classes which are not enabled within that subsystem. The warning and critical storage class thresholds are also activated by the MPS.

6.7.3. Gatekeeper

Each Gatekeeper may provide sites with the ability to:

- Monitor or control the use of HPSS resources using Gatekeeping Services.
- Validate user accounts using the Account Validation Service.

If the site doesn't want either service, then it is not necessary to configure a Gatekeeper into the HPSS system.

Sites can choose to configure zero (0) or more Gatekeepers per HPSS system. Gatekeepers are associated with storage subsystems. Each storage subsystem can have zero or one Gatekeeper associated with it and each Gatekeeper can support one or more storage subsystems. Gatekeepers are associated with storage subsystems using the *Storage Subsystem Configuration* screen (see the Storage Subsystem Configuration window section). If a storage subsystem has no Gatekeeper, then the Gatekeeper field will be blank. A single Gatekeeper can be associated with every storage subsystem, a group of storage subsystems, or one storage subsystem. A storage subsystem can *not* use more than one Gatekeeper.

Every Gatekeeper has the ability to supply the Account Validation Services. A bypass flag in the accounting policy metadata indicates whether or not account validation for an HPSS system is on or off. Each Gatekeeper will read the accounting policy metadata file, so if multiple Gatekeepers are configured and account validation has been turned on, then any Gatekeeper can be chosen by the Core Server to fulfill account validation requests.

Every Gatekeeper has the ability to supply the Gatekeeping Service. The Gatekeeping Service provides a mechanism for HPSS to communicate information through a well-defined interface to a policy software module to be completely written by the site. The site policy code is placed in a well-defined site shared library for the gatekeeping policy (either `/usr/local/lib64/libgksite.[a|so]` or `/opt/hpss/lib/libgksite.[a|so]`) which is linked to the Gatekeeper. The gatekeeping policy shared library contains a default policy which does *no* gatekeeping. Sites will need to enhance this library to implement local policy rules if they wish to monitor or load-balance requests.

The gatekeeping site policy code will determine which types of requests it wants to monitor (authorized caller, create, open, and stage). Upon initialization, each Core Server will look for a Gatekeeper in the storage subsystem metadata. If no Gatekeeper is configured for a particular

storage subsystem, then the Core Server in that storage subsystem will not attempt to connect to any Gatekeeper. If a Gatekeeper is configured for the storage subsystem that the Core Server is configured for, then the Core Server will query the Gatekeeper asking for the monitor types by calling a particular Gatekeeping Service API. This API will then call the appropriate Site Interface which each site can provide to determine which types of requests are to be monitored. This query by the Core Server will occur each time the Core Server (re)connects to the Gatekeeper. The Core Server will need to (re)connect to the Gatekeeper whenever the Core Server or Gatekeeper is restarted. Thus if a site wants to change the types of requests it is monitoring, then it will need to restart the Gatekeeper and Core Server.

If a Gatekeeper is configured, then it will either need to be running or marked non-executable for HPSS Client API requests to succeed in the Core Server (even if no gatekeeping or account validation is occurring); this is due to the HPSS Client API performing internal accounting initialization.

If multiple Gatekeepers are configured for gatekeeping, then the Core Server that controls the files being monitored will contact the Gatekeeper that is located in the same storage subsystem. Conversely, if one Gatekeeper is configured for gatekeeping for all storage subsystems, then each Core Server will contact the same Gatekeeper.

A Gatekeeper registers five different interfaces: Gatekeeper Service (also know as the *Functional Interface*), Account Validation Services, Administrative Services, Connection Manager Services, and Real-Time Monitoring Services. When the Gatekeeper initializes, it registers each separate interface. The Gatekeeper-specific configuration will contain any pertinent data about each interface.

The Gatekeeper Service interface provides the Gatekeeping APIs which call the site-implemented Site Interfaces. The Account Validation Service interface provides the Account Validation APIs. The Administrative Service provides the server APIs used by SSM for viewing, monitoring, and setting server attributes. The Connection Manager Service provides the HPSS connection management interfaces. The Real-Time Monitoring Service interface provides the Real-Time Monitoring APIs.

The Gatekeeper Service Site Interfaces provide a site the mechanism to create local policy on how to throttle or deny create, open and stage requests and which of these request types to monitor. For example, it might limit the number of files a user has opened at one time; or it might deny all create requests from a particular host or user. The Site Interfaces will be located in a shared library that is linked into the Gatekeeper.

It is important that the Site Interfaces return a status in a timely fashion. Create, open, and stage requests from MPS are timing sensitive, so the Site Interfaces won't be permitted to delay or deny these requests; however, the Site Interfaces may choose to be involved in keeping statistics on these requests by monitoring requests from authorized callers.

If a Gatekeeper should become heavily loaded, additional Gatekeepers can be configured (maximum of one Gatekeeper per storage subsystem). In order to keep the Gatekeepers simple and fast, they do not share state information. Thus if a site wrote a policy to allow each host a maximum of 20 creates, then that host would be allowed to create 20 files on each storage subsystem that has a separate Gatekeeper.

The Gatekeeper's Real-Time Monitoring Interface supports clients such as a Real-Time Monitoring

utility which requests information about particular user files or HPSS Request IDs.

6.7.4. PVL

The PVL is responsible for mounting and dismounting PVs (such as tape and magnetic disk) and queuing mount requests when required drives and media are in use. The PVL usually receives requests from Core Server clients. The PVL accomplishes any physical movement of media that might be necessary by making requests to the appropriate Physical Volume Repository (PVR). The PVL communicates directly with HPSS Movers in order to verify media labels. The PVL will manage the creation, deletion, and update of drives/devices within the HPSS system.

Only one PVL per HPSS system is supported.

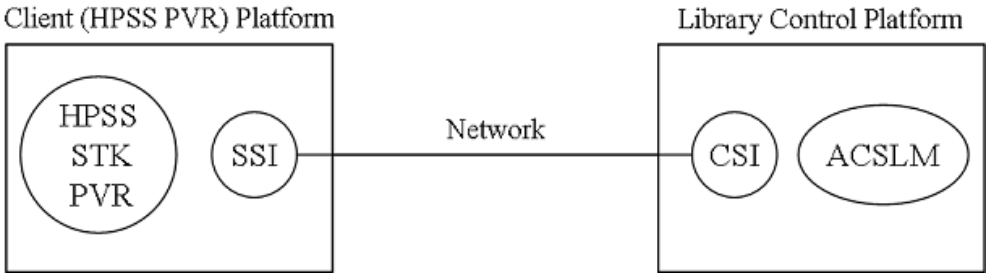
6.7.5. PVR

The PVR manages a set of imported cartridges and mounts and dismounts them when requested by the PVL. It is possible for multiple HPSS PVRs to manage a single library. This is done if it is necessary to organize the tape drives in the library into partitions. Each tape drive in the library is assigned to exactly one PVR. Additionally, each cartridge is assigned to only one PVR. The PVRs can be configured identically and can communicate with the library through the same interface.

The following sections describe the considerations for the various types of PVRs supported by HPSS.

6.7.5.1. STK PVR

The STK PVR communicates to the Oracle StorageTek ACSLS server via its Storage Server Interface (SSI) client software. The client software, maintained by Oracle Corp. can be obtained through HPSS support; see below and the STK PVR additional information section for additional information. The SSI client software must be running on the same machine as the HPSS STK PVR. The SSI client communicates with the Client System Interface (CSI) server via RPCs.



The SSI must be started before the PVR. If the SSI is started after the PVR, the PVR should be stopped and restarted.

If multiple Oracle StorageTek libraries are managed, SSIs that communicate with each of the library units should be configured on separate CPUs. A PVR can be configured on each of the CPUs that is running an SSI. If multiple Oracle StorageTek robots are connected and are controlled by a single Library Management Unit (LMU), a single PVR can manage the collection of robots. The PVR can be configured on any CPU that is running an SSI.

ACSLS should be running on the platform directly connected to the Oracle StorageTek library. The HPSS STK PVR can run on any platform that has a TCP/IP connection to the ACSLS platform. The

platform running the HPSS STK PVR must also be running Oracle StorageTek's SSI client software. This software will not be started by HPSS and should be running when HPSS is started. It is recommended that the SSI be started by the platform's initialization scripts every time the platform is booted.

The SSI requires that the system environment variables CSI_HOSTNAME and ACSAPI_PACKET_VERSION be correctly set. Note that due to limitations in the Oracle StorageTek Client System Component Developer's Toolkit, if the SSI is not running when the HPSS STK PVR is started, or if the SSI crashes while the HPSS STK PVR is running, the HPSS STK PVR will lock up and will have to be manually terminated by issuing `"kill -9 <pid>"`.

6.7.5.2. Operator PVR

The Operator PVR displays mount requests for manually mounted drives. The mount requests are displayed on the appropriate SSM screen.

All of the drives in a single Operator PVR must be of the same type. Multiple operator PVRs can be configured without any additional considerations.

6.7.5.3. SCSI PVR

The SCSI PVR communicates with tape libraries and robots through a generic SCSI interface. The interface uses the SCSI-3 command set.

6.7.6. Mover

The Mover configuration is largely dictated by the hardware configuration of the HPSS system. Each Mover can handle both disk and tape devices and must run on the node to which the storage devices are attached. The Mover is also capable of supporting multiple data transfer mechanisms (such as TCP/IP and shared memory) for sending data to or receiving data from HPSS clients.

6.7.6.1. Tape devices

All tape devices that will be used to read and write HPSS user data must be set to handle variable block sizes to allow for the ANSI standard 80-byte volume label and file section headers. This section describes the procedure for setting this option on each supported operating system.

HPSS supports tape devices on Linux with the use of the native SCSI tape device driver (**st**). To enable the loading of the Linux native tape device, uncomment the following lines in the `.config` file and follow the procedure for rebuilding your Linux kernel.

```
CONFIG_SCSI=y
CONFIG_CHR_DEV_ST=y
```

In Linux, tape device files are dynamically mapped to SCSI IDs/LUNs on your SCSI bus. The mapping allocates devices consecutively for each LUN of each device on each SCSI bus found at the time of the SCSI scan, beginning at the lower LUNs/IDs/buses. The tape device file will be in this format: `/dev/st[0-31]`. This will be the device name to use when configuring the HPSS device.

6.7.6.2. Disk devices

All locally attached magnetic disk devices (for example, SCSI or SSA) should be configured using the pathname of the raw device (that is, character special file).

For Linux systems, this may involve special consideration.

HPSS supports disk devices on Linux with the use of the native SCSI disk device driver (**sd**) and the raw device driver (**raw**).

The Linux SCSI Disk Driver presents disk devices to the user as device files with the following naming convention: `/dev/sd[a-h][0-8]`. The first variable is a letter denoting the physical drive, and the second is a number denoting the partition on that physical drive. Occasionally, the partition number will be left off when the device corresponds to the whole drive. Drives can be partitioned using the Linux **fdisk** utility.

The Linux raw device driver is used to bind a Linux raw character device to a block device. Any block device may be used.

See the Linux manual page for more information on the SCSI Disk Driver, the Raw Device Driver, and the **fdisk** utility.

To enable the loading of the Linux native SCSI disk device, uncomment the following lines in the configuration file and follow the procedure for rebuilding your Linux kernel.

```
CONFIG_SCSI=y
CONFIG_BLK_DEV_SD=y
```

Also, depending on the type of SCSI host bus adapter (HBA) that will be used, you will need to enable one or more of the lower level SCSI drivers. For example, if you are using one of the Adaptec HBAs with a 7000 series chipset, uncomment the following lines in the `.config` file and follow the procedure for rebuilding your Linux kernel.

```
CONFIG_SCSI_AIC7XXX=y
CONFIG_AIC7XXX_CMDS_PER_DEVICE=253
CONFIG_AIC7XXX_RESET_DELAY_MS=15000
```

6.7.6.3. Performance

The configuration of the Movers and attached devices can have a large impact on the performance of HPSS because of constraints imposed by a number of factors; for example, device channel bandwidth, network bandwidth, and processor power.

A number of conditions can influence the number of Movers configured and the specific configuration of those Movers:

- Each Mover process is built to handle a specific device interface, for example, IBM SCSI-attached 3590/3590H/3580 drives. If multiple types of devices are to be supported, multiple Movers must

be configured.

- Each Mover currently limits the number of concurrently outstanding connections. If a large number of concurrent requests are anticipated on the drives planned for a single Mover, the device workload should be split across multiple Movers. This is primarily an issue for Movers that will support disk devices.
- The planned device allocation should be examined to verify that the device allocated to a single node will not overload that node's resource to the point that the full transfer rates of the device cannot be achieved (based on the anticipated storage system usage). To offload a single node, some number of the devices and their corresponding Mover can be allocated to other nodes.
- In general, the connectivity between the nodes on which the Movers will run and the nodes on which the clients will run should have an impact on the planned Mover configuration. For TCP/IP data transfers, the only functional requirement is that routes exist between the clients and Movers; however, the existing routes and network types will be important to the performance of client I/O operations.
- Mover to Mover data transfers, performed for migration, staging, and repack operations, also impact the Mover configuration. For devices that support storage classes involved in migration or staging, the Movers controlling those devices should be configured so that there is an efficient data path among them. If Movers involved in a data transfer are configured on the same node, the transfer will occur via a shared memory segment.
- Additionally, the default scheduling of disk Mover requests is to spread I/O load evenly across the Movers. This assumes roughly equal bandwidth capacity for each Mover. This is often the simplest approach in planning, but in some cases some Movers may be configured with more or less bandwidth capacity than other Movers. In this case, the administrator can set `HPSS_SS_SCHEDULE_BY_CONTROLLER` in `env.conf` to `TRUE`. When `HPSS_SS_SCHEDULE_BY_CONTROLLER` is set, the Core Server will balance the disk devices by the disk controller ID. The controller ID is a settable field in the disk configuration. Disk resources which have shared infrastructure (e.g. disk controllers) can be set to the same controller ID to cause the Core Server to treat them as a unit. Alternatively, individual disk devices with no shared limitations can each have unique controller IDs so that they are scheduled independently. The Core Server then considers the controller assignment rather than the Mover assignment, which enables the Core Server to properly balance I/O requests across imbalanced Mover configurations.

6.7.7. Logging service

Logging Services are included in HPSS servers and tools.

HPSS servers and tools send log messages based on their log policies to syslog on the same node as the process. Based on the policy, messages may also be forwarded to the SSM for display in the Alarm and Events list.

6.7.8. Startup Daemon

The Startup Daemon is responsible for starting, monitoring, and stopping the HPSS servers. The daemon responds only to requests from the SSM System Manager. It shares responsibility with each HPSS server for ensuring that only one copy of the server runs at a given time. It helps the SSM determine whether servers are still running, and it allows the SSM to send signals to servers.

Normally, the SSM stops servers by communicating directly with them but, in special cases, the SSM can instruct the Startup Daemon to send a **SIGKILL** signal to cause the server to shut down immediately.

If a server is configured to be restarted automatically, the Startup Daemon will restart the server when it terminates abnormally. The daemon can be configured to restart the server without limit, or up to a fixed number of restarts, or not at all.

Choose a descriptive name for the daemon that includes the name of the computer where the daemon will be running. For example, if the daemon will be running on a computer named tardis, use the descriptive name "Startup Daemon (**tardis**)".

The Startup Daemon is started by running the `/opt/hpss/bin/rc.hpss` script. It ignores most signals and may only be killed using the **kill -9 <pid>** command. The Startup Daemon must be run under the *root* account so that it has sufficient privileges to start the HPSS servers.

The Startup Daemon runs on every HPSS server node. However, it does not run on remote Mover nodes.

6.7.9. Storage System Management

SSM has five components:

- A System Manager (**hpss_ssmsm**), which communicates with all other HPSS components requiring monitoring or control.
- The GUI (hpssgui(7)), which provides the HPSS administrator or operator the ability to configure or monitor the HPSS system through a set of windows.
- A Command Line Interface (hpssadm(7)), which provides the HPSS administrator or operator the ability to configure or monitor a subset of the HPSS system through a set of interactive or batch commands.
- An API (Quick startup of the SSM API), which provides the HPSS administrator or operator with a programmable interface.
- A web interface (Quick startup of the Web SSM UI), which provides HPSS operators with an easy web-interface to SSM statistics in any browser.

There can be only one SSM System Manager configured for an HPSS installation. The System Manager (SM) is able to handle multiple SSM GUI, Command Line, or API clients (on different hosts or on the same host).

Starting up the SSM GUI (hpssgui(7)) directly from the HPSS server node where the SSM System Manager is running and displaying the SSM window on the user's desktop is discouraged. This is due to known Java/X performance problems. Instead, it is recommended to install the Java and HPSS GUI client software on the user's desktop and execute it there. See the SSM desktop client packaging section for more information.

There are no performance problems associated with running the SSM Command Line Interface (hpssadm(7)) directly on the server UNIX platform, and this is the recommended configuration.

Both the SSM GUI client, hpssgui(7), and the SSM Command Line client, hpssadm(7), may be executed on any platform that complies with SSM client requirements.

Before starting the SM, a review of SM key environment variable settings would be wise. Following is a table of key SM environment variables along with the default value and meaning. Depending on the size (number of servers and number of SSM clients) and activity of the HPSS system, these values may need to be overridden in `env.conf`.

Table 9. Key SM environment variables

Variable	Default value	Functionality
HPSS_SM_SRV_ CONNECT_FAIL_COUNT	3	Connection fail count. The number of connection failures to a server before the maximum connection interval takes effect.
HPSS_SM_SRV_ CONNECT_INTERVAL_MIN	20	Interval between attempting server connections when the connection fail count has not yet been reached (in seconds).
HPSS_SM_SRV_ CONNECT_INTERVAL_MAX	60	Maximum connection interval. The interval between server connections when the connection fail count has been reached without a successful connection (in seconds).
HPSS_SM_SRV_ MONITOR_THREADS	5	The number of threads created to monitor server connections.
HPSS_SM_SRV_ QUEUE_SIZE	5	Request queue size used by the System Manager server interface. A default of five slots in the server interface request queue to be used when the server interface thread pool is completely full. The queue is used to hold RPC requests from servers until a thread is available to process the request. Note that if the request queue has any entries in it, it means that all the threads in the server thread pool are busy and the SM response will be degraded. If this happens, then it would be good to increase the number of threads available to the server interface using the HPSS_SM_SRV_TPOOL_SIZE variable. Increasing the size of the queue will not help with performance.

Variable	Default value	Functionality
HPSS_SM_SRV_ TPOOL_SIZE	100	Thread pool size used by the System Manager server interface. If the thread pool is exhausted, then server RPC requests will be queued in the server RPC request queue to wait for a thread to become available. When the thread pool is exhausted, SM performance may be degraded. Increase this value if that is the case. Typically, one thread per HPSS server should be adequate, but a few extra wouldn't hurt.
HPSS_SM_SRV_ MAX_CONNECTIONS	50	Number of HPSS server connections to maintain at once. If this number of connections is exceeded, then old connections will be closed to maintain this number of connections.

The SM attempts to throttle the connection attempts to other servers. It will attempt to reconnect to each server every HPSS_SM_SRV_CONNECT_INTERVAL_MIN seconds until the number of failures for that server has reached HPSS_SM_SRV_CONNECT_FAIL_COUNT. After the failure count has been reached the SM will only try to reconnect to the server every HPSS_SM_SRV_CONNECT_INTERVAL_MAX seconds until a successful connection is made at which time the connection interval for the server will be set back to HPSS_SM_SRV_CONNECT_INTERVAL_MIN.

6.8. Storage subsystem considerations

Storage subsystems are provided in HPSS for the purpose of increasing the scalability of the system, particularly with respect to the Core Servers. An HPSS system consists of one or more subsystems, and each subsystem contains its own Core Server. If multiple Core Servers are desired, this is accomplished by configuring multiple subsystems.

Each subsystem uses a separate DB2 subsystem database. Adding a subsystem to an HPSS system means adding an additional database that must be maintained and backed up. All subsystems share the config database.

6.9. Storage policy considerations

This section describes the various policies that control the operation of the HPSS system.

6.9.1. Migration policy

The migration policy provides the capability for HPSS to migrate (copy) data from one level in a hierarchy to one or more lower levels. The migration policy defines the amount of data and the

conditions under which it is migrated; however, the number of copies and the location of those copies is determined by the storage hierarchy definition. The site administrator will need to monitor the usage of the storage classes being migrated and adjust both the migration and purge policies to obtain the desired results.

6.9.1.1. Migration policy for disk

Disk migration in HPSS copies files from a disk storage class to one or more lower levels in the storage hierarchy. Removing or purging of the files from the disk storage class is controlled by the purge policy. The migration and purge policies work in conjunction to maintain sufficient storage space in the disk storage class.

When data is copied from the disk, the copied data will be marked purgeable but will not be deleted. Data is deleted by running purge on the storage class. If duplicate copies are created, the copied data is not marked purgeable until all copies have been successfully created. The migration policy and purge policy associated with a disk storage class must be set up to provide sufficient free space to deal with the demand for storage. This involves setting the parameters in the migration policy to migrate a sufficient number of files and setting the purge policy to reclaim enough of this disk space to provide the free space desired for users of the disk storage class.

Disk migration is controlled by several parameters. By default, these parameters are the same across all subsystems. However, subsystem-specific policies may be created which override all of these values. For a list of these parameters, refer to the Disk Migration Policy configuration section.

6.9.1.2. Migration policy for tape

The tape file migration feature allows a Class of Service to be configured so that files written to a tape storage class will be copied to another tape storage class and retained in both. This provides a backup copy of the files on tape.

The migration policy parameters which apply to tape file migration are detailed in the Tape Migration Policy configuration section.

6.9.2. Purge policy

The purge policy allows the MPS to remove the bitfiles from disk after the bitfiles have been migrated to a lower level of storage in the hierarchy. A purge policy should not be defined for a tape storage class or a disk storage class which does not support migration. Sites may or may not wish to define a purge policy for all disk storage classes that support migration. Purging from tapes is controlled by the "Migrate Files and Purge" flag of the tape migration policy; there is no separate purge policy for tape storage classes.

The specification of the purge policy in the storage class configuration enables the MPS to do the disk purging according to the purge policy for that particular storage class. Purge is run for a storage class on a demand basis. The MPS maintains current information on total space and free space in a storage class by periodically extracting this information from the HPSS Core Server. Based upon parameters in the purge policy, a purge run will be started when appropriate. The administrator can also force the start of a purge run via SSM.

The disk purge is controlled by several parameters:

- The **Do not purge files accessed within <nnn> minutes** parameter determines the minimum amount of time a site wants to keep a file on disk. Files that have been accessed within this time interval are not candidates for purge.
- The **Start purge when space used reaches <nnn> percent** parameter allows the amount of free space that is maintained in a disk storage class to be controlled. A purge run will be started for this storage class when the total space used in this class exceeds this value.
- The **Stop purge when space used falls to <nnn> percent** parameter allows the amount of free space that is maintained in a disk storage class to be controlled. The purge run will attempt to create this amount of free space. Once this target is reached, the purge run will end.
- The **Purge Locks expire after <nnn> minutes** parameter allows the length of time a file can be "purge locked" before it will appear on the MPS report to be controlled. The "purge lock" is used to prevent a file from being purged from the highest level of a hierarchy. Purge locks only apply to a hierarchy containing a disk on the highest level. HPSS will not automatically unlock locked files after they expire. HPSS reports the fact that they have expired in the MPS report.
- The **Purge by** list box allows sites to choose the criteria used in selecting files for purge. By default, files are selected for purge based on their migration time. Alternately, the selection of files for purging may be based on the time the file was created or the time the file was last accessed. Files may be purged in an unpredictable order if this parameter is changed while there are existing purge records already in metadata until those existing files are processed.

Purging is also controlled by the presence of super purge locks. These should be rare. The super purge lock prevents any level of the file from being purged except manually by force. It is set temporarily by the recover(7) utility and released when the recover(7) utility determines that a safe copy of the file still exists. It is set internally by the core server whenever a manual force purge is performed on a file, which should not be done except with guidance from HPSS support. Super purge locks do not expire. They may be removed manually by the **scrub purgeunlock** command with the assistance of HPSS support. The MPS and the plu(7) utility have not yet been updated to report super purge locks.

Administrators should perform performance tuning to determine the parameter settings that will fit the needs of their site. If a site has a large amount of disk file write activity, the administrator may want to have more free space and more frequent purge runs. However, if a site has a large amount of file read activity, the administrator may want to have smaller disk free space and less frequent purge runs, and allow files to stay on disk for a longer time.

6.9.3. Accounting policy and validation

The purpose of the accounting policy is to describe how a site will charge for storage, and, in addition, to describe the level of user authorization (validation) to be performed when maintaining accounting information. The policy is required for HPSS to run, even if the site doesn't charge for HPSS usage. The accounting information is extremely useful for understanding usage patterns and planning site layout and configuration. Even if sites do not charge for HPSS usage, the accounting information should be examined regularly. This information can assist sites in adjusting their system configuration to better handle their workload.

A site must decide which style of accounting to use before creating any HPSS files or directories. There are two styles of accounting: UNIX-style accounting and Site-style accounting. In addition, a

site may decide to customize their style of accounting by writing an accounting site policy module for the Gatekeeper.

If a site chooses Site-style accounting and account validation is turned off, LDAP must be used as the authorization mechanism. The hpssGECOS field which is maintained for each user in LDAP contains the account index which allows Site-style accounting to be used. However, if account validation is turned on, then the account index comes from the account validation metadata (through the Gatekeeper).

If UNIX authorization is used and account validation is turned off, UNIX-style accounting must be used because there is no hpssGECOS field. The basic limitation is that if the account index is needed out of the hpssGECOS field, it does not exist in UNIX. It only exists in LDAP.

The metadata for each file and directory in an HPSS system contains an Account field, which determines how the storage will be charged. Each user has at least one default account index, which is put into the Account field of all new files and directories.

When using UNIX-style accounting, the account index is the user's UID. When the user's UID is combined with the user's realm ID, a unique account is created.

When using Site-style accounting, each user may have more than one account index and may switch among them at runtime.

Each site must decide whether they wish to validate accounts. However, when using UNIX-style accounting no authorization checking need be done since the account is always the user's UID.

If account validation is enabled, additional authorization checks are performed when the following events occur: when files and directories are created, when their ownership is changed, when their account index is changed, or when a user attempts to use an account index other than their default. If the authorization check fails, the operation fails with a permission error.

Using account validation is highly recommended for sites that will be accessing remote HPSS systems. The use of account validation will help keep account indexes consistent. If remote sites are not being accessed, account validation is still recommended as a mechanism to keep consistent accounting information.

For Site-style accounting, an account validation metadata file must be created, populated, and maintained with valid user account indexes. See the Account Validation Editor (hpss_avaledit(7)) for details on the use of the Account Validation Editor.

If the **Require Default Account** field is enabled when using Site-style accounting and account validation, users are required to have valid default account indexes before performing almost any client API action. If the **Require Default Account** field is disabled (which is the default behavior) users will only be required to have a valid account set when performing an operation which requires an account to be validated such as a create, an account change operation, or an ownership change operation.

When using Site-style accounting with account validation, if the **Account Inheritance** field is enabled, newly created files and directories will automatically inherit their account index from their parent directory. The account indexes can then be changed explicitly by users. This is useful

when individual users have not had default accounts set up for them or if entire directory trees need to be charged to the same account. When **Account Inheritance** is disabled (which is the default) newly created files and directories will obtain their account from the user's current session account, which is initially set to the user's default account index. This default account index may be changed by the user during the session.

A site may decide to customize the way they do accounting. In most cases, these sites should enable account validation with Site-style accounting and then implement their own site policy module which will be linked with the Gatekeeper. See Gatekeeper as well as the appropriate sections of the *HPSS Programmer's Reference* for more information.

By default account validation is disabled (bypassed). If it is disabled, the style of accounting is determined by looking up each user's hpssGECOS account information in the authorization registry. The following instructions describe how to set up users in this case.

If users have their default account index encoded in a string of the form **AA=<default-acct-idx>** in the principal's LDAP hpssGECOS attribute, then Site-style accounting will be used; otherwise, UNIX-style accounting will be used.

To keep the accounting information consistent, it is important to set up all users in the HPSS Authorization services with the same style of accounting (that is, they should all have the **AA=** string in their hpssGECOS attribute or none should have this string). The **hpss_ldap_admin** tool can be used to set attributes for a user including the hpssGECOS field. For more information, see the **hpss_ldap_admin** man page.

See the Accounting section for more information.

6.9.4. Security policy

HPSS server authentication and authorization make extensive use of UNIX or Kerberos authentication and either UNIX or LDAP authorization mechanisms. Each HPSS server has configuration information that determines the type and level of services available to that server. HPSS software uses these services to determine the caller's identity and credentials. Server security configuration is discussed in more detail in the Server configuration section.

Once the identity and credential information of a client has been obtained, HPSS servers enforce access to their interfaces based on permissions granted by an access control list stored in the DB2 table AUTHZACL.

HPSS client interface authentication and authorization security features for end users depend on the interface and are discussed in the following subsections.

6.9.4.1. Client API

The Client API interface uses either UNIX username/password or Kerberos authentication and either UNIX or LDAP authorization features. Applications that make direct Client API calls must have valid credentials prior to making those calls. Kerberos credentials can be obtained either at the command line level via the **kinit** mechanism or within the application via the **sec_login_set_context** interface. UNIX credentials are determined by the HPSS rpc library based on the UNIX User ID and Group ID of the application process.

6.9.4.2. FTP/PFTP

By default, FTP and Parallel FTP (PFTP) interfaces use either a username/password mechanism or Kerberos credentials to authenticate. Either UNIX or LDAP is used to authorize end-users. The end user identity credentials are obtained from the principal and account records in the appropriate security registry.

6.9.4.3. Name space

Enforcement of access to HPSS name space objects is the responsibility of the Core Server. A user's access rights to a specific name space object are determined from the information contained in the object's ACL and the user's credentials.

6.9.4.4. Security audit

HPSS provides the ability to record information about authentication, file creation, deletion, access, and authorization events. The security audit policy in each HPSS server determines what audit records a server will generate. In general, all servers can create authentication events, but only the Core Server will generate file events. The security audit records are sent to HPSS logging services and recorded as security-type log messages.

6.9.5. Logging policy

The logging policy provides the capability to control which message types are written to the HPSS log files. In addition, the logging policy is used to control whether alarm and event messages are sent to the Storage System Manager to be displayed. Logging policy may be set on a per-server basis, or a default policy may be used for servers that don't have their own logging policy. Refer to the Creating a log policy section for a description of the supported message types.

If a logging policy is not explicitly defined for a server, the default log policy will be applied. The default log policy is selected from the *Global Configuration* window. If no default log policy entry has been defined, only Alarm and Event messages will be logged. All Alarm and Event messages generated by the server will also be sent to the Storage System Manager, if enabled.

The administrator might consider changing a server's logging policy under one of the following circumstances:

- A particular server is generating excessive messages. Under this circumstance, the administrator could use the logging policy to limit the message types being logged or sent to the Storage System Manager. This will improve performance and potentially eliminate clutter from the HPSS *Alarms and Events* window. Message types to disable first would be Trace messages followed by Request messages.
- One or more servers are experiencing problems which require additional information to troubleshoot. If Alarm, Debug, or Request message types were previously disabled, enabling these message types will provide additional information to help diagnose the problem. HPSS support personnel might also request that Trace messages be enabled for logging.

6.9.6. Gatekeeping

The Gatekeeping Service provides a mechanism for HPSS to communicate information through a well-defined interface to an installation specific customized software policy module. The policy module is placed in a shared library, `/usr/local/lib64/libgksite.[a|so]` or `/opt/hpss/lib/libgksite.[a|so]`, which is linked into the Gatekeeper. A module in `/usr/local/lib64` will take precedence over one placed in `/opt/hpss/lib` and can be useful for maintaining a site library across HPSS releases and installations. The default policy module does no gatekeeping. If gatekeeping services are desired in an HPSS installation, this default policy module must be replaced with one that implements the desired policy.

The site implemented policy module determines which types of requests will be monitored (authorized caller, create, open, and stage). Upon initialization, each Core Server looks for a Gatekeeper configured in its storage subsystem. If one is found, the Core Server asks the Gatekeeper for its monitor types by calling the **gk_GetMonitorTypes** API which calls the site implemented **gk_site_GetMonitorTypes** function which determines which types of requests to monitor. This query by the Core Server occurs each time the Core Server connects to the Gatekeeper, which occurs whenever the Core Server or Gatekeeper is restarted. Therefore, if a site wants to change the types of requests to be monitored, the Core Server and Gatekeeper must be restarted.

For each type of request being monitored, the Core Server calls the appropriate Gatekeeping Service API (**gk_Create**, **gk_Open**, **gk_Stage**) passing along information pertaining to the request. This information includes:

Table 10. Gatekeeping call parameters

Name	Description	create	open	stage
AuthorizedCaller	Whether or not the request is from an authorized caller. These requests cannot be delayed or denied by the site policy.	Y	Y	Y
BitFileID	The unique identifier for the file.	N/A	Y	Y
ClientConnectId	The end client's connection UUID.	Y	Y	Y
RealmId	The HPSS realm identifier for the user.	Y	Y	Y
GroupId	The user's group identifier.	Y	Y	Y
HostAddr	Socket information for originating host.	Y	Y	Y
OpenInfo	Open file status flag (Oflag).	N/A	Y	N/A

Name	Description	create	open	stage
StageInfo	Information specific to stage (flags, length, offset, and storage level).	N/A	N/A	Y
UserId	The user's identifier.	Y	Y	Y

Each Gatekeeping Service API will then call the appropriate Site Interface passing along the information pertaining to the request. If the request had AuthorizedCaller set to `TRUE`, then the Site "Stat" Interface will be called (**gk_site_CreateStats**, **gk_site_OpenStats**, **gk_site_StageStats**) and the Site Interface will not be permitted to return any errors on these requests. Otherwise, if AuthorizedCaller is set to `FALSE`, then the normal Site Interface will be called (**gk_site_Create**, **gk_site_Open**, **gk_site_Stage**) and the Site Interface will be allowed to return no error or return an error to either retry the request later or deny the request. When the request is being completed or aborted, the appropriate Site Interface will be called (**gk_site_Close**, **gk_site_CreateComplete**, **gk_site_StageComplete**). Examples of when a request gets aborted are when the Core Server goes **DOWN** or when the user application is aborted.

NOTES:

1. All open requests to the Core Server will call the Gatekeeping Service open API (**gk_Open**). This includes opens that end up invoking a stage.
2. Any stage call that is invoked on behalf of open will *not* call the Gatekeeping Service stage API (**gk_Stage**). For example, the ftp **site stage <filename>** command will use the Gatekeeping Service open API, **gk_Open**, rather than the Gatekeeping Service stage API, **gk_Stage**.
3. Direct calls to stage (**hpss_Stage**, **hpss_StageCallBack**) will call the Gatekeeping Service stage API (**gk_Stage**).
4. If the site is monitoring authorized caller requests, then the site policy interface won't be allowed to deny or delay these requests; however, it will still be allowed to monitor these requests. For example, if a site is monitoring authorized caller and open requests, then the site **gk_site_Open** interface will be called for open requests from users and the **gk_site_OpenStats** interface will be called for open requests due an authorized caller request (for example, migration by the MPS). The site policy can *not* return an error for the open due to migration; however, it can keep track of the count of opens by authorized callers to possibly be used in determining policy for open requests by regular users. Authorized caller requests are determined by the Core Server and are requests for special services for MPS. These services rely on timely responses, thus gatekeeping is not allowed to deny or delay these special types of requests.
5. The Client API uses environment variables HPSS_API_TOTAL_DELAY to place a maximum limit on the number of seconds a call will delay because of HPSS_ERETRY status codes returned from the Gatekeeper. See the Client API configuration section for more information.

Refer to *HPSS Programmer's Reference* for further specifications and guidelines on implementing the Site Interfaces.

6.10. Storage characteristics considerations

This section defines key concepts of HPSS storage and the impact the concepts have on HPSS configuration and operation. These concepts, in addition to the policies described above, have a significant impact on the usability of HPSS.

Before an HPSS system can be used, the administrator must create a description of how the system is to be viewed by the HPSS software. This process consists of learning as much about the intended and desired usage of the system as possible from the HPSS users and then using this information to determine HPSS hardware requirements and the configuration of the hardware to provide the desired performance. The process of organizing the available hardware into a desired configuration results in the creation of a number of HPSS metadata objects. The primary objects created are classes of service, storage hierarchies, and storage classes.

A storage class is used by HPSS to define the basic characteristics of storage media. These characteristics include the media type (the make and model), the media block size (the length of each basic block of data on the media), the transfer rate, and the size of media volumes. These are the physical characteristics of the media. Individual media volumes described in a storage class are called Physical Volumes (PVs) in HPSS.

Storage classes also define the way in which Physical Volumes are grouped to form Virtual Volumes (VVs). Each VV contains one or more PVs. The VV characteristics described by a storage class include the VV Block Size and VV data stripe width. RAIT tape storage classes also define the parity stripe width, the Minimum Write Parity, and the Read Verification flag. If PVs are grouped one at a time, so that their data stripe width is one, they are still defined as VVs.

A number of additional parameters are defined in storage classes. These include migration and purge policies, minimum and maximum storage segment sizes, and warning thresholds.

An HPSS storage hierarchy consists of multiple levels of storage where each level is described by a storage class. Files are moved up and down the storage hierarchy via stage and migrate operations, based upon storage policy, usage patterns, storage availability, and user requests. If a file is recorded at multiple levels in the hierarchy, the more recent data will be found at the higher level (lowest level number) in the hierarchy.

Class of Service (COS) is an abstraction of storage system characteristics that allows HPSS users to select a particular type of service based on performance, space, and functionality requirements. Each COS describes a desired service in terms of characteristics such as minimum and maximum file size, transfer rate, access frequency, latency, and valid read or write operations. A file resides in a particular COS which is selected when the file is created. Underlying a COS is a storage hierarchy that describes how data for files in that class are to be stored in the HPSS system. A COS can be associated with a fileset such that all files created in the fileset will use the same COS.

The relationship between storage class, storage hierarchy, and COS is shown in Relationship of Class of Service, storage hierarchy, and storage class.

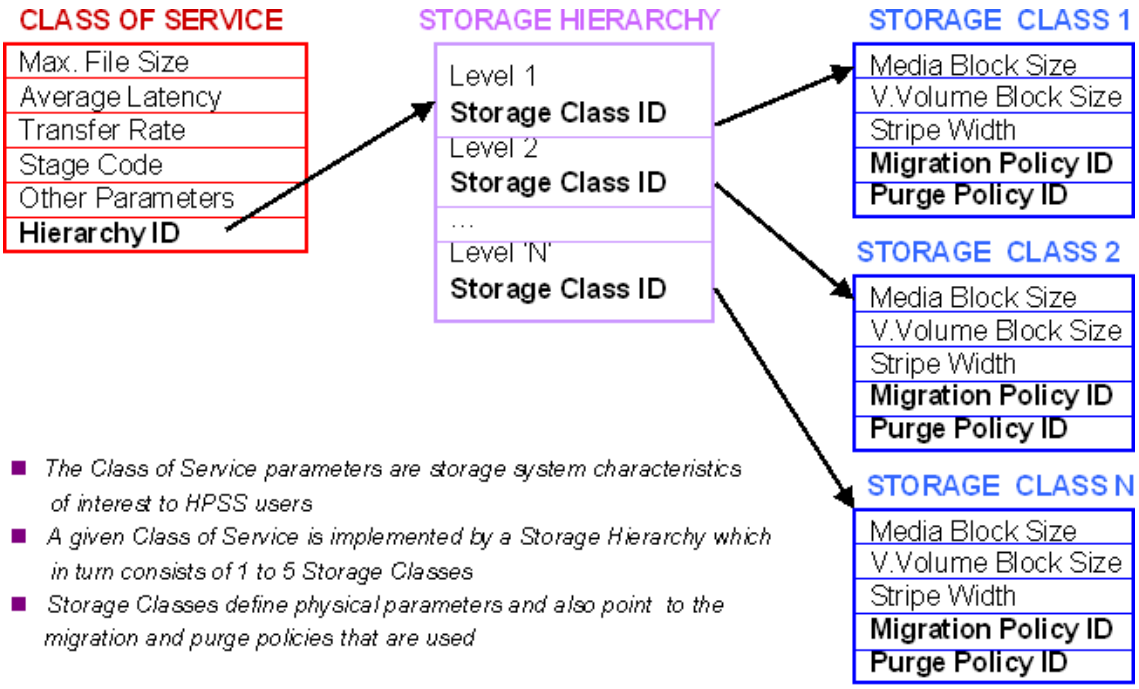


Figure 9. Relationship of Class of Service, storage hierarchy, and storage class

6.10.1. Storage class

Each virtual volume and its associated physical volumes belong to some storage class in HPSS. The SSM provides the capability to define storage classes and to add and delete virtual volumes to and from the defined storage classes. A storage class is identified by a storage class ID and its associated attributes. For detailed descriptions of each attribute associated with a storage class, see the Configured Storage Classes window section.

The sections that follow give guidelines and explanations for creating and managing storage classes.

6.10.1.1. Media block size selection

Guideline: Select a block size that is smaller than or equal to the maximum physical block size that a device driver can handle.

Explanation: For example, see Some recommended parameter values for supported storage media for recommended values for tape media supported by HPSS.

6.10.1.2. Virtual volume block size selection (disk)

Guideline: The virtual volume block size must be a multiple of the underlying media block size.

Explanation: This is needed for the correct operation of striped I/O. It may be necessary to experiment with combinations of disk and tape VV block sizes to find combinations that provide maximum transfer performance.

6.10.1.3. Virtual volume block size selection (tape)

Guideline 1: The VV block size must be a multiple of the media block size.

Explanation: This is needed for the correct operation of striped I/O.

Guideline 2: Pick an I/O transfer buffer size such that the size of the buffer being used to write this storage class is an integer multiple of the VV block size.

Explanation: Assume files are being written via standard FTP directly into a tape storage class. Also assume FTP is set up to use a 4 MB buffer size to write the data. This means that writes are done to the tape with a single 4 MB chunk being written on each write operation. If the tape virtual volume block size is not picked as indicated by the guideline, two undesirable things will happen. A short block will be written on tape for each one of these writes, which will waste data storage space, and the Core Server will create a separate storage segment for the data associated with each write, which wastes metadata space. Performing these extra steps will degrade transfer performance. See also Some recommended parameter values for supported storage media for further information about selecting block sizes.

Guideline 3: Disk and tape VV block sizes should be equal if possible.

Explanation: The system is designed to maximize the throughput of data when it is migrated from disk to tape or tape to disk. For best results, the sizes of the VV blocks on disk and tape in a migration path should be the same. If they are different, the data will still be migrated, but the Movers will be forced to reorganize the data into different size VV blocks which can significantly impact performance.

6.10.1.4. Stripe width selection

Stripe width determines how many physical volumes will be accessed in parallel when doing read/writes to a storage class.

Guideline 1: For tapes, the stripe width should be less than half the number of available tape drives. In the case of RAIT tapes, the total stripe width (data plus parity) should be less than half of the number of available tape drives.

Explanation: There must be enough tape drives to support the total stripe width. The repack(7) and recover(7) utility programs copy data from one tape VV to another, so the number of available tape drives of the appropriate type must be at least twice the total tape stripe width for these programs to function. Migration of files between tape storage classes in a hierarchy, that are of the same media type, requires at least twice as many available tape drives as the total stripe width of the storage class.

Guideline 2: Select a stripe width that results in data transmission rates to and from the drives matching or exceeding the network transmission rate.

Explanation: Configuring stripe widths that result in transmission rates that exceed the network transmission rate will waste device resources, since more hardware and memory (for Mover data buffers) will be allocated to the transfer, without achieving any performance improvement over a smaller stripe width. Also, if a large number of concurrent transfers are expected, it may be better, from an overall system throughput point of view, to use stripe widths that provide something less than the throughput supported by the network. The aggregate throughput of multiple concurrent requests will saturate the network. Overall throughput will be improved by consuming fewer device and memory resources. Note that when considering RAIT tapes, only the data stripe width

should be taken into consideration. Parity information is generated or consumed by the RAIT Engines and doesn't flow across the network with the data.

Guideline 3: For smaller files, use a small tape stripe width or a stripe width of "1".

Explanation: If writing directly to tape, rather than via a disk cache, writing a file will result in the mounting and positioning of all of the tapes before data transmission can begin. This latency will be driven by how many mounts can be done in parallel, plus the mount time for each physical volume. If the file being transmitted is small, all of this latency could cause performance to be worse than if no striping were used at all.

As an example of how to determine data stripe width based on file size and drive performance, imagine a tape drive that can transmit data at about 10 MB/second and it takes about 20 seconds on average to mount and position a tape. For a one-wide stripe, the time to transmit a file would be:

```
(<File Size in MB>/10) + 20
```

Now consider a 2-wide stripe for this storage class which has only one robot. Also, assume that this robot has no capability to do parallel mounts. In this case, the transmission time would be:

```
(<File Size in MB>/20) + (2 × 20)
```

The calculation indicates that the single stripe would generally perform better for files that are less than 400 MB in size.

Writing small files to RAIT tapes involves different considerations. The smallest RAIT virtual volumes are 2+1 - two data tapes and one parity tape. When writing these volumes, all three must be mounted, positioned, and written to, regardless of the size of the file. While this configuration may take more time to write than a 1-wide tape VV, the data is more secure. The failure of a single tape can be recovered by repacking the VV, while the failure of a single 1-wide data tape will take all of the files on the tape with it.

Recording small files twice, once on each of two levels of a Class Of Service, provides a backup copy of any given file if the primary copy fails. Secondary copies can be reconstructed from the primary copy as well. But this recording strategy consumes twice as much tape as one copy of the files occupies. The tape media utilization rate is 50%.

Recording the same files to a 2+1 RAIT storage class records the files one time with the same level of backup (one tape can be lost without data loss), but the tape utilization rate is 66%.

Guideline 4: Migration can use larger stripe widths.

Explanation: The tape virtual volume is usually mounted and positioned only once when migrating from disk to tape. In this case, larger stripe widths can perform much better than smaller stripe widths.

Guideline 5: The number of drives available for media in this storage class should be a multiple of the total stripe width.

Explanation: Unless the drives are shared across storage classes (which is usually the case), if the number of drives available is not a multiple of the total stripe width then less-than-optimal use of the drives is likely.

6.10.1.5. Blocks between tape marks selection (tape only)

The number of tape physical blocks written between tape marks can be controlled. Tape marks are generated for two reasons:

1. to force tape controller buffers to flush so that the Mover can better determine what was actually written to tape, and
2. to quicken positioning for partial file accesses.

In general, larger values for **Blocks Between Tape Marks** are favored as modern tape drives rely on data streaming to maximize performance. Aggregation of small files into relatively large streams of data on tape is an effective way to reduce the number of tape marks and increase migration rates. For recommended values for various media types, see Some recommended parameter values for supported storage media.

6.10.1.6. Minimum storage segment size selection (disk only)

The Core Server maps disk files onto a series of disk storage segments. The size of the storage segments is controlled by parameters from both the storage class configuration and the Class of Service configuration. It is determined by the **Min Storage Segment Size**, the **Max Storage Segment Size**, and the **Average Number of Segments** parameters from the storage class configuration and by the **Allocation Method** and **Truncate Final Segment** parameters from the Class of Service configuration. See the Class of Service Configuration window section for a description of how these parameters work together to determine the segment size. The smallest amount of disk storage that can be allocated to a file is determined by the **Min Storage Segment Size** parameter. This parameter should be chosen with disk space utilization in mind. For example, if writing a 4 KB file into a storage class where the storage segment size is 1024 KB, then 1020 KB of the space will be wasted. At the other extreme, each file can use at most 10,000 disk storage segments, so it isn't possible to write a terabyte file to a disk storage class with a maximum storage segment size below 128 MB. Under the Classic Style **Allocation Method**, when file size information is available, the Core Server will attempt to choose an optimal storage segment size between **Min Storage Segment Size** and **Max Storage Segment Size** with the goal of creating an **Average Number of Segments** for the bitfile.

Guideline 1: Care should be taken when selecting the minimum storage segment size. If data will be migrated from disks in the storage class to a tape storage class, the value of the **Minimum Storage Segment Size** parameter should meet one of the following conditions. These rules help prevent the creation of excessive numbers of tape storage segments when files are migrated from disk to tape.

- If the storage segment size is larger than the tape stripe length, it should be an integer multiple of the tape stripe length. The storage segment size may be equal to the tape stripe length.
- If the storage segment size is smaller than the tape stripe length, the tape stripe length should be an integer multiple of the storage segment size, and, it should be not more than 32 times the

storage segment size.

Guideline 2: When a large range of file sizes are to be stored on disk, define multiple disk storage classes with appropriate storage segment sizes for the sizes of the files that are expected to be stored in each storage class or consider using the Variable Length Allocation Method.

Explanation: The Class of Service (COS) mechanism can be used to place files in the appropriate place. Note that although the Core Server provides the ability to use COS selection, current HPSS interfaces only take advantage of this in two cases. First, the **pput** command in PFTP automatically takes advantage of this by selecting a COS based on the size of the file. If the FTP implementation on the client side supports the **alloc** command, a COS can also be selected based on file size. Files can also be directed to a particular COS with FTP and PFTP commands by using the **site setcos** command to select a COS before the files are stored. When setting up Classes of Service for disk hierarchies, take into account both the **Minimum Storage Segment Size** parameter and the **Maximum Storage Segment Size** parameter in determining what range of file sizes a particular COS will be configured for.

6.10.1.7. Maximum storage segment size selection

Maximum storage segment size selection (disk only)

This parameter, along with **Min Storage Segment Size** and **Average Number of Storage Segments** parameters from the storage class configuration and the **Allocation Method** and **Truncate Final Segment** parameters from the Class of Service configuration, is used by the Core Server to optimally choose a storage segment size for bitfiles on disk. The largest storage segment size that can be selected for a file in a storage class is limited by this parameter.

Guideline: In order to avoid creating excessive fragmentation of the space on disks in this storage class, it is recommended that this be set no higher than 5% of the size of the smallest disk Virtual Volume allocated in this storage class.

Maximum storage segment size selection (tape only)

The **Max Storage Segment Size** field in the storage class definition allows a site to limit the maximum amount of data written into a single tape storage segment. This can be useful when storing very large files that might span multiple tapes or fill an entire tape. By limiting the storage segment size, a site can take steps to guarantee that repack will be able to copy the storage segment to another tape during repack operations.

Repack requires that an existing tape storage segment be copied to a single target tape. A value of zero in this field indicates that tape storage segments are not limited in size. A nonzero value for this field must be between 10% and 50% of the VVSize defined for the storage class.

6.10.1.8. Maximum VVs to write (tape only)

This parameter restricts the number of tape VVs, per storage class, that can be concurrently written by the Core Server. It is meant to be used in "direct to tape" systems in which user files are written directly to tapes. Its purpose is to limit the number of tapes that can be simultaneously written so that a sudden increase in the number of clients writing tapes doesn’t cause the system to attempt to mount and write a large number of tapes. The number of tape drives used to write files in the

storage class will be limited to approximately the value of this field times the total stripe width defined for the storage class. Note that this field affects only tape write operations. Read operations are not limited by this parameter.

6.10.1.9. Average number of storage segments (disk only)

Under the Classic Style **Allocation Method**, this parameter, along with **Min Storage Segment Size** and **Max Storage Segment Size**, is used by the Core Server to optimally choose a storage segment size for bitfiles on disk. The Core Server attempts to choose a storage segment size between **Min Storage Segment Size** and **Max Storage Segment Size** that would result in creating the number of segments indicated by this field.

Guideline: For best results, it is recommended that small values (less than "10") be used. This results in minimizing metadata and optimizing migration performance. The default of "4" will be appropriate in most situations.

6.10.1.10. PV estimated size and PV size selection

Guideline: For tape, select a value that represents how much space can be expected to be written to a physical volume in this storage class with hardware data compression factored in.

Explanation: The Core Server uses this value as a guide in selecting tapes for writing, but regards it as an estimate only. Regardless of its value, the tape will be filled before another tape is chosen for writing.

Rule 1: For disk, the **PV Size** value must be the exact number of bytes available to be written on the PV. This value must be a multiple of the media block size and the VV block size. It may be necessary to round the actual size of the volume down to one of these multiples. The SSM will enforce these rules when the window fields are filled in.

Rule 2: For disk, the **PV Size** value must be less than or equal to the **Capacity** value described in the Configure a new device and drive section.

6.10.1.11. Optimum access size selection

Guideline: Generally, a good value for **Optimum Access Size** is the stripe length, which is the virtual volume block size times the data stripe width.

Explanation: This field is advisory in nature in the current HPSS release. In the future, it may be used to determine buffer sizes. Generally, a good value for this field is the stripe length; however, in certain cases, it may be better to use a buffer that is an integer multiple of the stripe length. The simplest thing at the present time is to set this field to the stripe length. It can be changed in the future without complication.

6.10.1.12. Some recommended parameter values for supported storage media

Suggested block sizes for disk and Suggested block sizes for tape contain suggested values for storage resource attributes based on the media type. The given values are not the *only* acceptable values, but represent reasonable settings for the various media types. See other subsections in Storage characteristics considerations for more information about setting the storage

characteristics.

Disk media parameters

The following table contains attributes settings for the supported disk storage media types.

Table 11. Suggested block sizes for disk

Disk type	Media block size	Minimum access size	Minimum virtual volume block size
SCSI attached	4 KB	0	1 MB
SAS	4 KB	0	1 MB
SSD	4 KB	0	1 MB
Fibre Channel attached	4 KB	0	1 MB

In the above table:

- "Media block size" is the block size to use in the storage class definition. For disk, this value should also be used when configuring the Mover devices that correspond to this media type. Note that this value will not limit the amount of data that can be read from or written to a disk in one operation; it is used primarily to perform block boundary checking to ensure that all device input/output requests are block aligned. This value should correspond to the physical block size of the disk device.
- "Minimum access size" is the size of the smallest access request that should regularly be satisfied by the media type. The performance of smaller accesses will be seriously degraded. A value of zero indicates that the media is suitable for supporting all data accesses.
- "Minimum virtual volume block size" is the smallest block size that should be configured for virtual volumes of the media type. Smaller values will cause increased overhead when transferring to or from volumes configured with stripe widths greater than one. Virtual volume block size has little or no effect on virtual volumes whose stripe width is one.

Note: *When SCSI, SAS, SSD, or Fibre Channel attached disks are combined to form striped virtual volumes, the minimum access size should become, at a minimum, the stripe width of the virtual volume multiplied by the virtual volume block size. If not, data access will only use a subset of the striped disks and therefore not take full advantage of the performance potential of the virtual volume.*

Tape media parameters

The following table contains attributes settings for the supported tape storage media types.

Table 12. Suggested block sizes for tape

Tape type	Media block size	Blocks between tape marks	Estimated physical volume size
IBM 3580 (LTO)	256 KB	2700	100 GB

Tape type	Media block size	Blocks between tape marks	Estimated physical volume size
IBM 3580 (LTO Gen 2)	256 KB	6300	200 GB
IBM 3580 (LTO Gen 3)	256 KB	14400	400 GB
IBM 3580 (LTO Gen 4)	256 KB	21600	800 GB
IBM 3580 (LTO Gen 5)	256 KB	25200	1.5 TB
IBM 3580 (LTO Gen 6)	512 KB	18000	2.5 TB
IBM 3580 (LTO Gen 7)	512 KB	27000	6.0 TB
IBM 3580 (LTO Gen 7 M8)	512 KB	27000	9.0 TB
IBM 3580 (LTO Gen 8)	512 KB	27000	12.0 TB
IBM 3580 (LTO Gen 9)	512 KB	36000	18.0 TB
IBM 3580 (LTO Gen 10)	512 KB	36000	30.0 TB
IBM 3580 (LTO Gen 10 Enterprise)	512 KB	36000	40.0 TB
IBM 3590	256 KB	1620	10, 20 GB
IBM 3590E	256 KB	2520	20, 40 GB
IBM 3590H	256 KB	2520	60, 120 GB
IBM 3592 J1A	256 KB	7200	60 GB (JJ/JR), 300 GB (JA/JW)
IBM 3592 E05 (TS1120)	256 KB	1800	100 GB (JJ/JR), 500 GB (JA/JW), 700 GB (JB/JX)
IBM 3592 E06 (TS1130)	256 KB	28800	128 GB (JJ/JR), 640 GB (JA/JW), 1000 GB (JB/JX)
IBM 3592 E07/EH7 (TS1140)	256 KB	43200	500 GB (JK), 1.6 TB (JB/JX), 4.0 TB (JC/JY)
IBM 3592 E08/EH8 (TS1150)	256 KB	54000	900 GB (JK), 7.0 TB (JC/JY)

Tape type	Media block size	Blocks between tape marks	Estimated physical volume size
IBM 3592 E08/EH8 (TS1150)	256 KB	64800	2.0 TB (JL), 10.0 TB (JD/JZ)
IBM 3592 55E/F/G (TS1155)	256 KB	64800	3.0 TB (JL), 15.0 TB (JD/JZ)
IBM 3592 60E/F/G (TS1160)	256 KB	72000	5.0 TB (JM), 20.0 TB (JE/JV)
IBM 3592 70F/S (TS1170)	256 KB	72000	50.0 TB (JF)
Sony GY-8240	256 KB	4320	60, 200 GB
Sony SAIT-1	256 KB	5400	500 GB
StorageTek 9840A	256 KB	1800	20 GB
StorageTek 9840B	256 KB	3600	20 GB
StorageTek 9840C	256 KB	5400	40 GB
StorageTek 9840D	256 KB	5400	75 GB
StorageTek 9940A	256 KB	1800	60 GB
StorageTek 9940B	256 KB	5400	200 GB
StorageTek T10000A	256 KB	21600	500 GB
StorageTek T10000B	256 KB	21600	1000 GB
StorageTek T10000C	256 KB	43200	5000 GB
StorageTek T10000D	256 KB	45720	8000 GB

In the above table:

- "Media block size" is the block size to use in the storage class definition. This is the size of the data blocks written to tape. Note that for tape devices, the Mover configuration does not contain the media block size. This value may have a significant impact on data transfer performance, as for most tape devices each input/output request must be for the media block size. If a large block size is used for relatively small write requests, space may be wasted on the tape.
- "Blocks between tape marks" is the number of media blocks to be written between tape marks. A relatively small value has the benefit of shorter positioning times to the middle of files. Small values have the penalties of poorer media utilization and lower performance when writing tapes. Since files are usually read in their entirety, and modern tape controllers employ sophisticated positioning logic and are designed to stream data to tape, larger values of the **Blocks Between Tape Mark** parameter are recommended. The values in the table above are guidelines that should provide good general performance. It is possible that better performance might be achieved via experimentation with this setting in your environment.

- "Estimated physical volume size" is the estimated size of the physical volumes to be set in the storage class definition. These values are based on the expected media to be used with the specified type. In some cases, different length tape media may be used, which may have an effect on the estimated size for a given physical volume (for example, regular or extended length 3480/3490 format cartridges). Note that the values listed do not take into account any data compression that may be performed by the tape drive. Also, note that this value is for informational purposes only and does not affect the amount of user data written to a tape volume by the Core Server. The server fills each tape Virtual Volume such that the amount of data written to the tape varies with the compressibility of the data.

6.10.2. Storage hierarchy

Each HPSS file is stored in a storage hierarchy consisting of an ordered list of storage classes. A storage hierarchy can have up to five levels starting with level 0. The highest level (first level written to) is always level 0 and the lowest is level 4. Storage classes are expected to be arranged in a hierarchy in order of descending performance. For example, a level 0 storage class could be a fast disk while a level 4 storage class could be a slower, large-capacity tape system. The SSM provides a means to define storage hierarchies. A storage hierarchy is identified by a storage hierarchy ID and its associated attributes. For detailed descriptions of each attribute associated with a storage hierarchy, see the Storage Hierarchy Configuration window section. The following is a list of rules and guidelines for creating and managing storage hierarchies.

Rule 1: All writes initiated by clients are directed to the highest level (level 0) in the hierarchy.

Rule 2: Parts or all of a file may appear at multiple levels in a storage hierarchy. If data for a file does appear at multiple levels of the hierarchy, the data at the higher level is always the more recent data.

Rule 3: Migration of data does not skip levels in the hierarchy, except in the special case of creating duplicate copies when doing disk migration.

Rule 4: The client stage command can only stage data to the top level (level 0) in the hierarchy.

Rule 5: A given storage class can only occur once in the same hierarchy.

6.10.3. Class of Service

Each HPSS file belongs to a single Class of Service (COS) which is selected when the file is created. It is selected via Class of Service hints information passed to the Core Server when the bitfile is created. If using the Client API, the application program has full access to this hints information. FTP users can use the **quote** command to set the COS. A **pput** request in PFTP automatically selects a COS based on file size unless the user explicitly selects the COS.

The SSM provides a means to define Classes of Service. A COS is identified by a COS ID and its associated attributes. For detailed descriptions of each attribute associated with a Class of Service, see the Class of Service Configuration window section.

The Force Selection flag can be set in the COS definition to prevent automatic selection. If this flag is set, the COS can only be selected by asking for the COS by ID or Name.

The sections that follow give guidelines and explanations for creating and managing classes of service.

6.10.3.1. Selecting minimum file size

Guideline: This field is used to indicate the smallest file that should be stored in the COS.

Explanation: This limit is not enforced and is advisory in nature. The minimum file size can be used as a criteria for selecting a COS via the COS hints mechanism. Currently, PFTP and FTP clients that support the **alloc** command will use the size hints when creating a new file. The SSM will enforce the requirement that the **Minimum File Size** is less than the **Maximum File Size**.

6.10.3.2. Selecting maximum file size

Guideline: This field can be used to indicate the largest file that may be stored in the COS.

Explanation: If the **Enforce Max File Size** option is selected, an attempt to perform an operation on a file that would cause this value to be exceeded will be rejected. The underlying storage hierarchy should be set up so that the defined storage classes support files of this size in a reasonable fashion. For details, see Storage class and Storage hierarchy on storage classes and storage hierarchies. This field can be used via the COS Hints mechanism to affect COS selection. PFTP and FTP clients that support the **alloc** command will use the size hints when creating a new file.

6.10.3.3. Selecting stage code

This field determines whether a file is to be staged to the highest level in the storage hierarchy when the file is opened by one of the Client API function calls to the Core Server. This field can be used via the COS Hints mechanism to affect COS selection. Stage behavior is also affected by the value of the Auto Stage Retry flag.

Guideline 1: Select the **No Stage** option if staging of files is not desired.

Explanation: Data read from the file may come from lower levels in the storage hierarchy if the data does not exist at the top level. This option is normally selected if the top level in the hierarchy is not disk or if the users of files stored in this COS wish to control staging directly via user stage requests.

Guideline 2: Select the **Stage on Open** option to stage the entire file to the top level of the hierarchy synchronously. The Client API open operation will block while the file is being staged.

Explanation: This option is commonly selected when the top level of the hierarchy is disk and the files in this Class of Service are small to moderate in size. Use this option if you want to be guaranteed that the file is completely and successfully staged before it is read. If the stage operation fails, the open will return with an error.

Guideline 3: Select the **Stage on Open Async** option if you wish to stage the entire file to the top level in the hierarchy and do not want the Client API open to block.

Explanation: When this option is selected, the file is staged in sections and the read and write operations that access this file are blocked only until the portion of the file they reference has been completely staged. Normally, this option is selected when the top level of the hierarchy is disk and the files in this COS are fairly large in size. This option is honored only when the top level in the

hierarchy is disk. If the top level is tape and this option is specified, the **No Stage** option will be used instead.

Guideline 4: Select the **Stage on Open Background** option if you want the stage operation to be queued in the Core Server and processed by a background Core Server thread.

Explanation: The Client API open request will return with success if the file is already staged. Otherwise, a stage request is placed in a queue and will be processed by the Core Server in the background. A busy error is returned to the caller. This option allows a large number of stages (up to 2000) to be queued in the Core Server and processed as resources permit. The other stage options will result with a busy error if Core Server threads are not immediately available to process the request.

Guideline 5: Select the **Auto Stage Retry flag** if you want to configure a storage class so that stage failures of the primary copy can be automatically retried from a valid second copy. The associated hierarchy must first be configured for multiple copies.

Explanation: When a stage from the primary copy fails and a secondary copy of the file is available, HPSS will usually reissue the stage operation from the secondary copy. This is typically done when a tape, holding the primary copy, becomes damaged and cannot be read. A warning that the stage operation has used the secondary copy will appear in the SSM *Alarms and Events* window.

6.10.3.4. Selecting optimum access size

This field is only advisory in nature; however, for later releases it may be used by interfaces to select good buffer sizes dynamically.

Guideline 1: Generally, if the file is being staged on open, **Optimum Access Size** should be set to the same value that **Optimum Access Size** is set to in the storage class at the top of the hierarchy.

Guideline 2: If data is not being staged to the top level before it is read (either automatically or by user command), select a value that is an integer multiple of the largest **Optimum Access Size** field found among the storage classes that make up this hierarchy.

6.10.3.5. Selecting average latency

This field can be used via the COS Hints mechanism to affect COS selection.

Guideline 1: This field should generally be set to the value of the **Average Latency** field in the storage class at the top level of the hierarchy when the **Stage on Open** option is in effect. If files are usually accessed only once after they are staged, the average latency should be set to the latency of the level they are staged from.

Guideline 2: If it is expected that most of the requests for files in this COS are read requests, then it may be best to set the value of this field equal to the **Average Latency** field in the Storage Class in the hierarchy where most of the data accesses come from.

Guideline 3: If files are written into the COS much more frequently than read, use the **Average Latency** field from the Storage Class at the top level in the hierarchy.

6.10.3.6. Selecting transfer rate

This field can be used via the COS Hints mechanism to affect COS selection.

Guideline 1: This field should generally be set to the value of the **Transfer Rate** field in the storage class that is at the top level in the hierarchy. This should always be the case if the data is being staged on open.

Guideline 2: If a large percentage of the reads are being done from a lower level in the hierarchy, consider setting the transfer rate based on the **Transfer Rate** associated with the storage class at this lower level.

6.10.3.7. StripeLength and StripeWidth hints

These fields can be used via the COS Hints mechanism to affect COS selection.

Guideline: StripeLength and StripeWidth hints are available in the hints mechanism. When specified in the hints, StripeLength and StripeWidth from the storage class at the top level of each hierarchy are used in the COS selection algorithm.

6.10.4. File families

A file family is an attribute of an HPSS file that is used to group a set of files on tape virtual volumes. When a file is migrated from disk to tape, it is migrated to a tape virtual volume assigned to the family associated with the file. If no tape is associated with a family, the file is migrated to the next available tape not associated with a family, then that tape is assigned to the family. The family affiliation is preserved when tapes are repacked. Each file in HPSS is assigned a family designation. The default family is family zero, which is interpreted by HPSS as no family affiliation. Configuring file families is an HPSS administrator function.

HPSS places no restriction on the values assigned to the File Family IDs, other than the special meaning of family zero. A name must be assigned to each file family.

To associate a file with a file family, the HPSS administrator could associate a file family with a fileset. This will cause all files created in the fileset to be assigned to a particular file family. Alternately, the HPSS administrator could allow the user to set the file family attribute of each file that they want assigned to a file family. This will need to be done just after creation, but before data transfer. See the user's guide for your file transfer applications for further information on the commands (for example, FTP supports **getfam** and **setfam** to get and set the file family for a file).

Defining multiple file families may have an impact on system migration performance. MPS may have to mount significantly more tapes to complete a migration from a disk storage class if the files are spread across a large number of file families, compared to the number of mounts that would be required if all the files were in the same family.

6.11. HPSS performance considerations

This section provides some additional considerations for enhancing HPSS performance.

6.11.1. DB2

Because all HPSS operations involve DB2, it is important to optimize DB2 performance. There are a number of configuration settings that can greatly affect DB2 performance. Some of these are discussed in Tune DB2. For a detailed discussion on tuning DB2, refer to the *Database Fundamentals >> Performance tuning* section under the DB2 InfoCenter.

The following is a list of areas to consider when optimizing DB2 performance for HPSS:

- Database usage
  - Average number of users and applications connected to DB2
  - Maximum users and applications connected to DB2
  - Nature of usage: read or update
- Database logging
  - Hardware or software mirroring
  - Disk speed and reliability: select the fastest, most reliable disk
  - To help achieve best availability and logging performance, separate the transaction logs and DB2 data, or tablespaces, onto separate spindles and onto separate LUNse
- Database recovery
  - Enabling dropped table recovery will decrease database performance
- Amount of available memory and size of buffer pools
  - Buffer pools should not be larger than the tables using them (that is, not over-allocating memory)
  - Increase buffer pool size (that is, increase memory available for metadata)
- Type of node
  - Amount of cache, memory, CPU available
- Scale Db2 transactions across multiple database partitions and distribute the partitions across multiple computers to run as many transactions in parallel as possible.
- Storage media

6.11.2. Bypassing potential bottlenecks

HPSS performance is influenced by many factors, such as device and network speeds, configuration and power of HPSS server nodes, DB2 configuration, storage resource configuration, and client data access behavior.

HPSS provides mechanisms to bypass potential data transfer performance bottlenecks, provided that the system configuration provides the additional resources necessary. For example, if the performance of a single disk device is the limiting factor in a transfer between HPSS and a client application, the disks can be reconfigured in a striped storage class to allow parallel transfers at higher transfer rates. If after forming the stripe group, the I/O or processor bandwidth of a single node becomes the limiting factor, the devices can be distributed among a number of nodes, alleviating the limitations of a single node.

If the client node or single network between the client and HPSS becomes the limiting factor, HPSS supports transferring data to or from multiple client nodes in parallel, potentially using multiple physical networks, to bypass those potential bottlenecks.

HPSS also supports a single client using multiple sockets per HPSS device in a transfer request. This can greatly improve individual transfer speeds. The method to enable this is client-specific. Consult the application man pages for details on how to enable this feature. For application developers, the **Simplified PIO Interface** documented in the **HPSS Programmers Reference** can provide an easy to use API for tools that want to add multiple socket support and improve performance.

During system planning, consideration should be given to the number and data rates of the devices, node I/O bandwidth, network bandwidth, and client node bandwidth to attempt to determine a configuration that will maximize HPSS performance given an anticipated client workload.

6.11.3. Configuration

The configuration of the HPSS storage resources (see Storage characteristics considerations) is also an important factor in overall HPSS performance, as well as how well the configuration of those resources matches the client data access patterns.

For example, if a site provides access to standard FTP clients and allows those clients to write data directly to tape, the buffer size used by the FTP server and the virtual volume block size defined for the storage class being written to will have a significant impact. If the buffer size used by the FTP server is not a multiple of the virtual volume block size, each buffer written will result in a distinct storage segment on the tape. This will cause additional metadata to be stored in the system and extra synchronization processing of the tape. However, if the buffer size is a multiple of the virtual volume block size, each write will continue to append to the same storage segment as the previous write. This will continue until the final write for the file, which will usually end the segment, thus reducing the amount of metadata generated and media processing.

6.11.4. FTP/PFTP

Data transfers performed using the standard FTP interface are primarily affected by the buffer size used by the FTP daemon. The buffer size can be configured as described in the FTP/PFTP daemon configuration section. It should be a multiple of the storage segment size, if possible. Otherwise, it should be at least a multiple of the virtual volume block size. If the buffer size is too small, the FTP daemon will need to issue a large number of individual read or write requests; however, if the buffer size is too large, the FTP daemon will require a large amount of memory, which may cause additional paging activity on the system.

The size of the FTP daemon buffer is extremely important if the FTP clients write files directly to a tape storage class, as described in Configuration.

Parallel FTP (PFTP) uses TCP/IP to move data.

Note that the PFTP data transfer commands (such as **pput** and **pget**) are not influenced by the FTP daemon buffer size because the data flows directly between the client and Movers.

Note that PFTP clients that use the standard FTP data transfer commands (such as **put** and **get**)

have the same performance considerations as standard FTP clients.

Parallel transfers move the data between the Mover and the end-client processes bypassing the HPSS FTPD. Users should be educated to use the parallel functions rather than the non-parallel functions.



ASCII transfers are not supported by the parallel functions and the non-parallel functions will need to be specified for ASCII transfers. ASCII transfers are *not* typically required, but the end-customer should familiarize themselves with the particulars.

Parallel transfers should be optimized so that the Class of Service (COS), media stripe widths, network stripe widths, and Parallel Block Sizes are consistent with each other. For example, using a network stripe width of "4" with a media width of "2" may result in poorer performance than if both specifications are equal. Specifying a network stripe width of "4" where there is only one network interface may not provide any improvement over a lower network stripe width ("2") if the bandwidth of the network is (over-)filled by a 4-way stripe.

Non-parallel transfers occur via a "stage and forward" approach (device <==> Mover <==> HPSS FTP daemon <==> FTP client.) It is recommended that the "Non-Parallel Hostname" option be specified in the `HPSS.conf` file if the FTP daemon system has multiple interfaces. The hostname should refer to the highest speed interface available for transmission of data between the FTP daemon and HPSS Movers.

Where reasonable, the standard FTP ports should be modified to something other than 20/21 on the system acting as the HPSS FTP daemon. The HPSS FTP daemon should be set to use the 20/21 ports by default. This reduces the problem of requiring the end-customer to know which port to use for transfers to HPSS. In conjunction with this, it is highly recommended that the {ftpbanner} file be used with an appropriate message to provide information to the end-customer that they are accessing HPSS as opposed to a standard system.

6.11.5. Client API

The Client API provides the capability to perform data transfer of any size (the size being parameters supplied by the client to the read and write interfaces). The size of the data transfers can have a significant impact on the performance of HPSS. In general, larger transfers will generate less overhead than a series of smaller transfers for the same total amount of data.

The size of the transfers is extremely important because the clients may write files directly to a tape storage class, as described in Configuration.

6.11.6. Core Server

Minor performance degradations may be seen when the Core Server is processing path names with a large number of components, servicing requests for objects which have a large number of Object ACL entries, and servicing create requests for objects which have Initial Container/Initial Object ACL entries.

1. *Location Service*

The Location Service itself will give warning when a problem is occurring by posting alarms to SSM. Obtain the information for the Location Service alarms listed in the *HPSS Error Manual*.

If the Core Server consistently reports a heavy load condition, increase the number of request threads and apply the config.

6.11.7. Logging

Excessive logging by the HPSS servers can degrade the overall performance of HPSS. If this is the case, it may be desirable to limit the message types that are being logged by particular servers. There are two ways to do this.

The most effective way of managing excessive logging also provides the least fine-grained control. The server Logging Policy can be updated to control which message types are logged. A default Log Policy may be specified to define which messages are logged. Typically, Trace, Accounting, and Request messages are not logged. Other message types can also be disabled. Once the Logging Policy is updated for one or more HPSS servers, it will begin being used the next time those servers scan to update their logging policies (usually within 30 seconds).

Another way to manage excessive logging, which provides more control, but less overall performance benefit, is to use syslog filtering. This can be done using normal syslog filtering mechanisms to prevent messages from being logged. However, HPSS still forwards the messages to syslog.

6.11.8. Cross-realm trust

Cross-realm trust should be established with the minimal reasonable set of cooperating partners. Excessive numbers of cross-realm connections may diminish security and cause performance problems due to Wide Area Network (WAN) delays. The communication paths between cooperating realms should be reliable.

6.11.9. Gatekeeping

Sites may choose to implement site policy in the Gatekeeper for load balancing create, open, and stage requests. The site policy could limit the maximum number of non-authorized caller requests allowed at once by either delaying or denying particular requests. To delay the request, the site policy may return a special retry status along with the number of seconds to wait before the Client API retries the request. Delaying requests should limit the number of create, open, and stage requests performed at a particular point in time, thus decreasing the load on the system. However, care must be taken to figure out the best retry wait scheme to meet the requirements for each site and to configure the correct number of Gatekeepers if the load on one Gatekeeper is heavy. (Note: The maximum number of Gatekeepers per storage subsystem is one.) Also, sites need to write their Site Interfaces optimally to return in a timely manner.

Two special error status codes (**HPSS_ETHRESHOLD_DENY** and **HPSS_EUSER_DENY**) may be used to refine how a site may deny a create, open, or stage request. If the Core Server receives either of these errors, then it will return this error directly to the Client API rather than performing a retry. Errors other than these two or the special **HPSS_ERETRY** status will be retried several times by the Core Server. See either volume of the *HPSS Programmer's Reference* for more information.

Create, open, and stage requests from authorized callers (MPS) can *not* be delayed or denied due to timing sensitivity of the special requests these servers make to the Core Server. For example, migration of a file by MPS is an authorized caller open request. The site policy could keep track of authorized caller requests to further limit non-authorized caller requests.

If a Gatekeeper is being used for Gatekeeping Services, then the Core Server for each storage subsystem configured to use a particular Gatekeeper will return errors for the create, open, and stage requests being monitored by that Gatekeeper when that Gatekeeper is down. For example, if storage subsystem #2 is configured to use Gatekeeper #2, and Gatekeeper #2 is monitoring open requests and is DOWN, then each open by the Core Server in storage subsystem #2 will eventually fail after retrying several times.

6.11.10. HPSSFS-FUSE interface

Refer to the *HPSSFS-FUSE Administrator's Guide*, bundled with the HPSSFS-FUSE RPM and hpssfs(7).

6.11.11. SSM API

The SSM API by default is tuned to run alongside other services on the core server and should not require any tuning for standard usage.

The SSM API can be tuned by modifying the `hpss_ssmapi.conf.py` file in `/var/hpss/etc` to control timeout values, maximum number of requests, and the maximum number of worker processes.

6.11.12. Web SSM UI

The Web SSM UI is a standalone web application. It does need to be able to contact an SSM API server in order to acquire SSM data to display, but that server is not required to be on the same system.

The Web SSM UI can be tuned by modifying the `hpss_webssm.conf.py` file in `/var/hpss/etc` to control timeout values, maximum number of requests, and the maximum number of worker processes.

6.12. HPSS metadata backup considerations

This section contains guidelines for proper maintenance of the HPSS metadata stored in DB2.



The policies described should be fully understood and implemented to protect the HPSS metadata. Failure to follow these policies can lead to unrecoverable data loss.

The remainder of this section is a set of rules associated with backing up HPSS metadata. Though automated archive software like Tivoli Storage Manager is used to backup and protect DB2 data, it is important that each site review this list of rules and check to ensure that their site's backup is consistent with these policies.

When deciding on the size and number of disks for metadata, keep in mind the following:

1. At a minimum, for subsystem-specific metadata, the disks hosting the DB2 instance installation path (that is, the home directory of the instance owner) and the databases' tablespaces should

be separate. This is not critical for configuration metadata since it changes infrequently.

2. Ideally, several physical disks should be available for DB2 tablespaces (for example, so that name space data can be on one and bitfile data on another).
3. If the backup software employed involves some sort of disk staging area, this should be on a separate disk, and a large amount of disk should be allocated for this purpose.

For reliability and availability reasons, the disk hosting the instance installation data should be mirrored. The disks hosting the tablespaces should also be mirrored if possible. For performance and reliability reasons, mirroring should be done using separate physical devices.

6.13. HPSS security considerations

The security requirements between sites differ widely. The HPSS system administrators must be aware of the sites security requirements and should be aware of the security configuration required in HPSS. Sites should contact HPSS support if they have questions regarding HPSS security. For more information on security, see the Security and system access section.

6.14. HPSS Disk Mover and VV Load-Balancing

6.14.1. Overview

The HPSS Core Server performs basic I/O load balancing across disk movers and disk VVs. It tracks ongoing allocations and read/write I/O jobs for each disk mover and disk VV. This allows the Core Server to have an in-the-moment sense of the number of I/O tasks currently being served by all of the disk devices in the HPSS system. The Core Server uses these metrics when deciding where to allocate new disk segments. It will avoid busier movers and disk VVs during allocation: in other words, the Core Server will bias allocations toward less-busy movers and disk VVs.

These disk operations metrics are useful not only for the Core Server to balance incoming I/O operations across the complement of disk movers, but also for HPSS system administrators to understand how/why the Core Server made its I/O load-balancing decisions. This can be useful for monitoring whether allocations and I/O are evenly balanced across all the disk VVs or whether there are any hot-spots of I/O activity. Hot-spots could be the result of unevenly balanced data across the disk VVs or unevenly distributed disk volumes. Learning about *how* I/O is load-balanced can provide opportunities for ensuring that an HPSS system's disk infrastructure is well-balanced.

6.14.2. Disk Mover & VV I/O Metrics

The Core Server's Storage Service (SS) maintains I/O metrics against each of the disk movers and the disk VVs that those movers manage. The I/O metrics indicate how many read or write I/O tasks are ongoing against the movers and their disk VVs. As an I/O task begins, the SS increments the *operations in progress* for the mover(s) and VV(s) involved in the I/O. When the I/O is complete, the SS decrements the *operations in progress* for the mover(s) and VV(s) involved in the I/O. Note that read I/O and write I/O are weighted the same: the metrics are incremented and decremented exactly the same way regardless of whether reading from or writing to a mover and disk VV.

The SS uses these metrics to guide disk allocations. At any point in time, the number of *operations in*

progress for the various disk movers and VVs give a point-in-time snapshot of how busy the movers and disks are. This is a simple measurement of how many I/O tasks are currently in flight against the movers and their VVs. At point of allocation, the SS finds the mover and disk VV with the lowest number of operations in progress and, if enough space is available, performs the allocation against that mover and disk VV. An allocation represents future I/O activity. Once the I/O for that allocation begins running, that's when the *operations in progress* will be incremented.

The *time last allocated* is used as a tie breaker if movers and their VVs have the same number of operations in progress. In the case of a tie breaker, the least recently allocated mover and VV will be chosen. Again, note that read and write operations are given the same weight. If **mover A** has 8 *operations in progress* (4 reads; 4 writes) and **mover B** has 8 *operations in progress* (8 reads; 0 writes), they'll be considered to be equivalently busy.

Disk allocations can be guided to the least busy mover and disk VV. This is how disk write I/O is load balanced across the available disk movers and disk VVs. However, disk reads are not load-balanced. Those I/O tasks must occur on the movers and disk VVs that hold the requested data. The read I/O can't be shifted to other movers and VVs. So an influx of disk read requests can lead to an I/O imbalance across the disk mover complex. Note that both stages from disk and migrations from disk to tape are considered to be disk read I/O activity.

The I/O metrics are ephemeral. They age out pretty quickly. The metrics are interesting close in time to when the allocations and I/O occur, especially during various testing and debugging efforts. But after a short amount of time the metrics are no longer interesting. So the in-memory metadata is regularly trimmed. This is an attempt to control how much memory is used by the I/O metrics, especially on busy production systems.

The disk mover and VV I/O operations metrics are available in two ways:

- To enable monitoring of the disk I/O operations metrics that the Core Server keeps in memory, the Core Server dumps the metrics in JSON format to files at a regular cadence. These files are regularly trimmed in order to ensure that they don't fill the file system where they are created. See the description in the next section for details.
- A point-in-time snapshot of the I/O metrics. This is available in the `showdiskmaps` tool's output. Directly below is an example for the 4-way VV DLD21200. Note that, as more concurrent I/O jobs run against HPSS, the `OpsInProgress` fields will increment.

```
VVID: Object: 000222e0-03-00000001-01e53bc3a4fcb612-35b1, Disk VV
      System ID: 140000, Server ID: 1
      DB2 format: x'000222E0030000000101E53BC3A4FCB61235B1'
PV Names: DLD21200 DLD21300 DLD21400 DLD21500
I/O Metrics
 ├ VV
 │  ├ OpsInProgress:     4
 │  └ TimeLastAllocated: 1629172518406154
 └ Mover ID 1041
   ├ OpsInProgress:     4
   └ TimeLastAllocated: 1629172518406154
Number of Extents: 11, Map State: MAP_FREE, Storage Class ID: 140, Flags: None
Block Size: 67108864 bytes, Number of Clusters: 56079
```

```
Usable Length:              15053323501568 bytes
Actual Length:              15053659045888 bytes
Free Space:                 15021111246848 bytes
Cluster Length:                 268435456 bytes
Number of Map Blocks: 18
Number of slots in use: 13
Number of levels in use: 11
```

6.14.3. Maintaining Disk I/O Operations Metrics Files

The files are stored in the path pointed to by the HPSS_PATH_TMP environment variable (or /var/hpss/tmp by default if that environment variable isn't set). The dump files can be identified by the following file name pattern: <hpss_core-PID>.disk-op-state.<seconds-since-UNIX-epoch>.json

By default, the disk I/O operations metrics files are created once per minute and trimmed once they're 15 minutes old. This means there should be around 15 of the files present in the file system at any given time. These values were chosen to allow the files to remain in the file system long enough to be consumed by monitoring systems and short enough that the file system isn't overwhelmed with many of these files. For HPSS systems with high disk I/O workloads, these files may become large because they represent every disk allocation and I/O task performed by the HPSS system.

You can control both the frequency at which these files are created and how old they're allowed to become before being trimmed. You may want to increase or decrease the age and creation frequency of these files depending on whether you want to monitor them, how often your monitoring system examines them, the I/O workload handled by your HPSS system, and the size of the file system in which these files are created. You can control the creation frequency and age of the dump files with these two environment variables:

HPSS_DISK_OP_METRICS_DUMP_INTERVAL_SECS

: The frequency, in seconds, at which a new disk I/O operations dump file is created. A value of 60 means once a minute. (default = 60)

HPSS_DISK_OP_METRICS_DUMP_FILE_AGE_SECS

: How old the dump files are allowed to be before being trimmed. Once the dump files hit the specified age in seconds, they are removed from the file system. A value of 900 means remove the files once they are 15 minutes old. (default = 900)

To turn off the automated output of the disk operations I/O metrics into these dump files, set either of the above interval or the file age environment variables to 0. You can still manually cause the Core Server to output disk I/O operations metrics to a JSON-formatted dump file by initiating a Core Server state dump. Follow the link for details about generating a server state dump.

The Core Server will start using new values for these environment variables after you reinitialize the Core Server.

6.14.4. JSON format

There are three flavors of JSON entries in the disk I/O operations metrics dump files: disk allocation, disk I/O begin, disk I/O completion.

6.14.4.1. Disk Allocation

This JSON entry for a disk allocation is unique from the other two types of metrics (I/O begin and completion). This type of entry shows which disk movers and volumes were considered for allocation. For the list of disk movers and volumes that were considered and rejected, a rejection reason is listed for each. The disk mover and volume that were chosen for the allocation are listed separately in the entry.

This type of entry includes, but is not limited to, information about:

- The mover and VV that were chosen for the allocation
- The time of the current and most recent allocation
- The other disk VVs for the chosen mover that were considered for allocation (along with a reason string for rejecting them)
- The other disk movers that were considered for allocation (along with the reason string for rejecting them)

The entry for a disk allocation looks like:


```
{
 "requestID": "c76a10231e05ea42b2392c58b4f69325",
 "timestamp": "2024-10-25T09:44:58.649271-0800",
 "type": "write",
 "moverAccepted": {
  "id": 8,
  "previousAllocation": "2024-10-25T09:42:35.618468-0800",
  "currentOperations": 0,
  "currentAllocation": "2024-10-25T09:44:58.764156-0800",
  "previousOperations": 0,
  "virtualVolumeAccepted": {
   "name": "DSK0001",
   "previousAllocation": "2024-10-25T09:42:35.618468-0800",
   "currentOperations": 0,
   "currentAllocation": "2024-10-25T09:44:58.351388-0800",
   "previousOperations": 0
  },
  "virtualVolumeRejections": [
   {
    "name": "DSK0002",
    "previousAllocation": "2024-10-25T09:44:02.852547-0800",
    "currentOperations": 0,
    "reason": "used too recently"
   },
   {
```

```
     "name": "DSK0003",
     "previousAllocation": "2024-10-25T09:43:08.325478-0800",
     "currentOperations": 0,
     "reason": "used too recently"
    },
    {
     "name": "DSK0004",
     "previousAllocation": "2024-10-25T09:42:58.125487-0800",
     "currentOperations": 0,
     "reason": "used too recently"
    }
   ]
  },
  "moverRejections": [
   {
    "id": 9,
    "previousAllocation": "2024-10-25T09:43:48.547886-0800",
    "currentOperations": 0,
    "reason": "used too recently"
   },
   {
    "id": 10,
    "previousAllocation": "2024-10-25T09:44:12.214785-0800",
    "currentOperations": 0,
    "reason": "used too recently"
   }
  ]
 }
```


6.14.4.2. Disk I/O Begin & Completion

In addition to tracking disk allocation operations, the disk I/O operations metrics track disk I/O begin and completion. A disk I/O operation begin entry is created just prior to issuing a read or write I/O request (an IOD, or I/O Descriptor) to the disk movers involved in the I/O request. A corresponding disk I/O operation completion entry is created when all the disk movers have responded back to the Core Server with the results (an IOR, or I/O Results) of the I/O request.

Note that pairs of begin and completion entries can be matched via the `requestID` field. There will be a balanced number of these entries for each `requestID`. The entry that comes first is the beginning of the I/O. The one that comes second is the completion of the I/O. You can also note that, when the I/O begins, the number of ongoing operations against the mover and VV increases by the stripe width of the I/O (stripe width of 4 in the above example). And the number of ongoing concurrent operations decreases in the the I/O completion entry.

These types of entries include, but are not limited to, information about:

- The mover and VV performing the disk I/O
- The I/O type (read or write)
- The number of concurrent I/O tasks being served by the mover and VV

- The timestamps of the current and most recent allocations

The disk I/O begin and completion entries look like the below:


```
{
 "requestID": "dad58bcd897ae246b6f4c4e9407880b8",
 "timestamp": "2024-10-25T09:56:56.946418-0800",
 "type": "write",
 "mover": {
  "id": 8,
  "currentAllocation": "2024-10-25T09:56:55.164875-0800",
  "previousOperations": 0,
  "currentOperations": 4
 },
 "virtualVolume": {
  "name": "0000002a-03-00000004-01ef92f22423a72c-0000",
  "currentAllocation": "2024-10-25T09:56:55.794354-0800",
  "previousOperations": 0,
  "currentOperations": 4
 }
},
{
 "requestID": "dad58bcd897ae246b6f4c4e9407880b8",
 "timestamp": "2024-10-25T09:56:56.749841-0800",
 "type": "write",
 "mover": {
  "id": 8,
  "currentAllocation": "2024-10-25T09:56:55.616887-0800",
  "previousOperations": 4,
  "currentOperations": 0
 },
 "virtualVolume": {
  "name": "0000002a-03-00000004-01ef92f22423a72c-0000",
  "currentAllocation": "2024-10-25T09:56:55.648613-0800",
  "previousOperations": 4,
  "currentOperations": 0
 }
}
```


6.14.5. Monitoring Disk I/O Operations Metrics

These metrics are expected to be most suitable for a time-series monitoring system such as Prometheus, time-series data stream models in Elasticsearch, and others. It should be possible to display these metrics in service of monitoring such things as:

- I/O hotspots
- ongoing load-balance of HPSS disk mover and VV infrastructure
- distribution of disk reads vs disk writes across the disk movers and disk VVs

A timestamp is encoded into the filename of each disk I/O operations metrics dump file. The format of each dump file's name is: `<hpss_core-PID>.disk-op-state.<seconds-since-UNIX-epoch>.json`

To enable monitoring in a time-series fashion, subseqent dump files contain the last dump file interval's worth of disk I/O operations. It's recommended that the monitoring system keep track of the the timestamp of the most recent dump file that it consumed and consume the next set of available dump files during the next monitoring sweep. The monitoring system needs to consume the dump at least as often as the `HPSS_DISK_OP_METRICS_DUMP_FILE_AGE_SECS` in order to not miss any metrics. Otherwise, there will be a time gap in the metrics displayed by the monitoring system.

Chapter 7. System preparation

This section covers the steps that must be taken to appropriately prepare your system for installation and configuration of HPSS and its infrastructure.

General setup

Set up file systems

Set up tape libraries

Verify tape drives

Set up disk drives

Set up network parameters

To understand and optimize the operation of HPSS, some baseline measurement, evaluation, and tuning of the underlying network and IO subsystems is necessary. Appropriate tuning of these components is necessary for HPSS to perform as expected. Any one component that is not performing at its optimal rate will have an adverse effect on the performance of the entire system. The steps and tools listed in this chapter will help a site make the best use of their available resources. The measurements taken should be saved for future reference. If performance problems occur later on, these values can be compared with new measurements to determine if the performance problems are related to changes in the subsystems. Though disk and tape configurations rarely change without an administrator's knowledge, networks can "mysteriously" degrade for a variety of reasons. This is especially true for client access to the HPSS system.

7.1. General setup

- Download copies of the *HPSS Admin Guide* for the appropriate version of HPSS. It will probably be useful to print copies of these documents and keep them handy while installing HPSS.
- Install, configure, and verify the correct prerequisite operating system version on all HPSS, Kerberos client and/or server, and DB2 nodes.
- Check the format of `/etc/hosts`. If `/etc/hosts` is used to augment DNS, fully qualified hostnames must be listed first. For example:

```
123.45.6.789 host1.domain.gov host1
```

- Verify the current operating system level:

```
% uname -a
```

- Create appropriate HPSS UNIX user accounts. The "hpss" user and group ID should be created at

this time.

- Install the prerequisite software for Kerberos, C compiler, Java, Perl, SSH, OIDC, HTTP server, and any other specific software that will be used by HPSS. Verify the correct patch levels are, or will be, installed. Refer to Prerequisite software considerations for additional information.
- Configure the Perl prerequisite software on HPSS nodes.
- Contact your HPSS Support representative to get a copy of the HPSS Test Plan (HTP) and install it on each node. Run and become familiar with the **lsnode** tool, which will be helpful in other steps. Additional support tools are provided on the HPSS Administrator Wiki (https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=start) under support → support_tools.

  To run **lsnode** and save the output to `/var/hpss/stats/lsnode.out`:

  ```
  % cd /<HTP_install_location>/config
  % ./lsnode.ksh > /var/hpss/stats/lsnode.out
  ```

- Obtain your HPSS Realm ID from HPSS support. Make sure that the Realm ID does not exceed 32 bits. This information will be needed when **mkhpss** is used to configure HPSS. For an existing installation, this is the ID which was previously referred to as the DCE Cell ID.

7.2. Set up file systems

The following sections describe how to set up the various file systems used by HPSS and DB2.

7.2.1. DB2 file system

Each database in DB2 has its own log. We recommend that these logs be placed on a file system reserved exclusively for this purpose. This should be a fast, but more importantly, stable disk, preferably a RAID device. For example, the following file systems might be created for a configuration database and subsystem database:

```
/db2_log/cfg
/db2_log/subsys1
```

The NEWLOGPATH database configuration variable can be used to direct the database logs to these file systems.

Additionally, the following file systems are required for storing the mirrored copy of the DB2 logs:

```
/db2_logmirror/cfg
/db2_logmirror/subsys1
```

Additionally, the following file systems are required for storing the archived copy of the DB2 logs:

```
/db2_logarchive1/cfg
```

```
/db2_logarchive1/subsys1
/db2_logarchive2/cfg
/db2_logarchive2/subsys1
```

The DB2 log mirroring is configured using the MIRRORLOGPATH variable.

The home directory of the DB2 instance owner should be created as a separate file system on each HPSS node that runs a database instance. By default, the DB2 instance owner is "hpssdb" and its home directory is `/db2data/db2_hpssdb`.

7.2.2. HPSS file system

Configure `/var/hpss` as a separate file system on each HPSS server node. This file system will store HPSS configuration files, log files, MPS reports, and other HPSS-related files. It is recommended that this file system be at least 1 GB in size.

The `/var/hpss/log` subdirectory will be created automatically and will contain HPSS log files after you run the `configure_logging` tool during installation (see Configure Logging).

Configure `/var/hpss/adm/core` as a separate file system on each HPSS server node. If an HPSS process fails and creates a core file, it will be placed in this location. It is recommended that this file system be configured with at least 2 GB of disk space on server nodes and 1 GB on Mover nodes.

Configure `/db2data/db2_hpssdb` as a separate file system on the HPSS core server node. This file system stores the HPSS DB2 instance configuration information as well as the *CFG* database tables. This file system should be created with at least 1 GB of disk storage and, like many of the other file systems related to HPSS, it should be monitored for fullness periodically by the administrator.

Configure `/opt/hpss` as a separate file system on each of the HPSS server and Mover nodes. This directory contains the HPSS executables, documentation, libraries, and, for those sites with source code, the development environment to build the HPSS system. For most sites, this file system can be configured with 1 GB of space. For those sites with source code, at least 2 GB are required.

7.3. Set up tape libraries

7.3.1. Oracle StorageTek

For an Oracle StorageTek tape library:

- If using an Oracle StorageTek tape library, configure the ACSLS (server) and SSI (client) software properly and verify that it is working correctly.

  To test the ability to mount and dismount a tape in an Oracle StorageTek library, use the **stk_ctl** utility.

  To mount a tape:

  ```
  % stk_ctl mount <driveSpec> <tapeLabel>
  ```

where **driveSpec** is four integers separated by commas (no spaces), identifying the ACS, LSM, panel, and drive (for example, "0,0,1,2").

To dismount a tape:

```
% stk_ctl dismount <driveSpec>
```

To query a drive:

```
% stk_ctl query <driveSpec>
```

Refer to the STK PVR-specific configuration window section for more information.

7.3.2. SCSI

For a SCSI-connected tape library, configure the SCSI Medium Changer (SMC) library device on the node that will run the HPSS SCSI PVR, and verify that it is operational.

To test the ability to mount and dismount a tape in a SCSI-connected library, use the **umccp** utility.

To start **umccp**:

```
%umccp <devname>
```

where **devname** is the SMC device for that library. The device names for SMCs can be determined using the device_scan(7) utility. **umccp** will inventory the library before prompting you for a command.

To mount a tape:

```
umccp:  mount <cartridge> <drive address>
```

where cartridge is a 6-character cartridge label and drive address is one of the addresses given by the **umccp drives** command.

To dismount a tape:

```
umccp:  dismount <drive address> [destination address]
```

If both parameters are used, the cartridge will be dismounted to the destination address. Otherwise, the first free address will be used.

To query a drive:

```
umccp:  query drive
```

This will query all drives.

Note that **umccp** does not modify its internal inventory as you perform operations. Use the **umccp inventory** command to update the inventory state.

Refer to the SCSI PVR-specific configuration window section for more information.

7.4. Verify tape drives

Verify that the correct number and type of tape devices are available on each tape Mover node.

Repeat the above steps for each tape drive.

7.4.1. Linux

On each tape Mover node, verify that each tape drive has the variable-length block size option enabled.

To determine if the variable block size option is enabled, the following should complete successfully:

```
% dd if=/dev/null of=/dev/rmt/tps2d6nr bs=80 count=1
% dd if=/dev/null of=/dev/rmt/tps2d6nr bs=1024 count=1
```

If the variable-length block size option is not enabled, consult your driver documentation for procedures to enable it.

On each tape Mover node, verify that the raw read and write I/O performance of all HPSS tape drives are at the expected levels. Create one or more tables documenting the results. The HPSS Test Plan (HTP) provides a baseline test for generating tape drive performance outputs.

To conduct the read and write tests on **st1**, mount a scratch tape on **st1** and issue the following commands.



The contents of this tape will be overwritten by iocheck, so be sure to mount the correct tape cartridge.

To measure uncompressed write performance (but note that specifying nst will cause the tape to not rewind):

```
% iocheck -w -t 20 -b 1mb /dev/nst1
```

To measure the maximum-compressed write performance on st1 (and then rewind the tape):

```
% iocheck -w -t 20 -f 0 -b 1mb /dev/st1
```

To measure read performance on drive st1 using the previously-written uncompressed and compressed files:

```
% iocheck -r -t 20 -b 1mb /dev/nst1
% iocheck -r -t 20 -b 1mb /dev/nst1
```

To empty the tape:

```
% mt -f /dev/st1 rewind
% mt -f /dev/st1 weof 2
```

7.5. Set up disk drives

7.5.1. Linux

For Linux platforms, specific commands and syntax are not listed in this document. Perform the following steps using the appropriate operating system commands:

- Verify that the correct number and type of disk devices are available on each DB2 and disk Mover node.
- Create all necessary raw disk volumes to be used by the HPSS disk Movers. If a file system interface is being used, then create the sparse files to be used as devices.
- On each disk Mover node, measure the raw read and write I/O performance of all HPSS disks and verify that they are at expected levels. Create one or more tables documenting the results. The output of these tests should be stored in `/var/hpss/stats` for later analysis.

7.6. Set up network parameters

- Install and configure all network interfaces and corresponding network connections.

  Refer to IBM's internal network technologies home page for resources on configuring and tuning networks and TCP/IP.

  The network interfaces section of the **lsnode** report from each node shows the network interfaces that are configured.

  To determine how many network interfaces are available (Linux):

  ```
  % netstat -i
  ```

  For interface information, use the Name column from **netstat -i** in the **ifconfig** commands. Note

the IP address follows the inet phrase in the output from the **ifconfig** command.

To test whether an IP address is reachable (nonzero exit status indicates the **ping** was not successful):

```
% ping -c 1 <ipAddress>
```

- Isolate Kerberos communication to the designated control networks on all HPSS and DB2 nodes in order to separate the HPSS control and data paths.
- Place all HPSS, DB2, and Kerberos server node IP addresses in a local host table (`/etc/hosts`).

  For Linux, changes should be made to `/etc/nsswitch.conf`:

  ```
  hosts: files nis dns
  ```

- Verify that network TCP throughput has been optimized and the performance of each network is at expected levels in both directions (especially check HPSS data networks between Movers and between Mover and client nodes). Using **iperf** or another network tool, measure the performance of all the networks that will be used in communications between Kerberos, HPSS, DB2, and client nodes. If multiple paths between nodes are to be used, then all of them need to be measured as well. The transfer rates for networks differ greatly depending upon the individual technology and network settings used. It is important to gather performance data using a variety of settings to determine the optimal combinations. The primary values that govern performance include send/receive buffers, size of reads and writes, and **rfc1323** value for high performance networks (such as Gigabit Ethernet). Create a table showing these values.

Consult the network tool documentation for measuring performance. The HPSS Test Plan (HTP) has a test which can be used for generating network performance results using **iperf**.

Each node's send and receive performance should be measured for each interface planned for use. Multiple block sizes should be tried to find the optimal settings for your network.

You are looking for the best values possible for each network connection. These values will be used by HPSS to optimize its data transfers. This example is, by no means, a complete picture of what controls network performance. In fact, it is assumed that you have already optimized the networks. The reason for gathering these values is to optimize HPSS performance on an already tuned network, not to fix underlying network problems.

HPSS makes extensive use of a system's networking capabilities. Therefore, the setting of the tunable networking parameters for the systems on which the various HPSS servers and clients will run can have a significant impact on overall system performance.

Some options that typically impact performance within an HPSS system environment are:

Table 13. Network options

Network option	Description
thewall	Controls the maximum amount of system memory that can be used by the system networking code. A value that is too low can cause networking requests to be delayed or denied.
sb_max	Controls the maximum size allowed for send and receive buffers for a socket.
udp_recvspace	Controls the default size of the receive buffer for UPD/IP sockets. A value that is too small can cause server RPC sockets to be overrun.
tcp_recvspace, tcp_sendspace	Controls the default size for the receive and send buffers for TCP/IP sockets. Internally, HPSS servers and clients attempt to set these buffers sizes explicitly, but other utilities may not.
rfc1323	Controls whether large TCP window sizes are used. Usually set to ON for higher throughput networks (such as 10 Gb Ethernet) and set to OFF for lower throughput networks (such as 10 Mb Ethernet).

It is recommended that the available combination of options be tested as part of the initial HPSS system testing. In particular, poor network performance has been experienced where options on one system do not match options on other remote systems.

There are also attributes that are specific to the individual network interface that may affect network performance. It is recommended that the available interface-specific documentation be referenced for more detailed information.

The anticipated load should also be taken into account when determining the appropriate network option settings. Options that provide optimal performance for one or a small number of transfers may not be the best settings for the final multi-user workload.

7.6.1. `HPSS.conf` configuration file

The `HPSS.conf` configuration file contains tuning options to be used by HPSS clients and servers. See the HPSS.conf(7) man page for details.

The `HPSS.conf` file may also contain options used by non-HPSS applications. Application developers are asked to observe the "Reserved for Future Use" components specified in Appendix D.

The `HPSS.conf` configuration file is located in the directory named by either:

- The **HPSS_CFG_FILE_PATH** environment variable,
- the directory `/usr/local/etc`, or

- the directory `/var/hpss/etc` (preferred),
- in that order. If the file is not present or no matching entry is found, the Parallel FTP Client, Client API, and Mover will use system defaults.

See Appendix D: `HPSS.conf` configuration file for more details.

7.7. Port mapping and firewall considerations

If your Core Server machines are behind a firewall and you wish to access your HPSS system from an application that uses the HPSS client library from the other side of the firewall, you will have to configure both HPSS and the firewall to allow for this. The HPSS client library will connect to the Core Servers and Gatekeeper Servers configured into your system. In order for this to work, you will have to configure HPSS so that these servers listen for connections on a (small) range of ports. You will also have to configure the firewall to allow incoming connections to this range of ports and to port 111 (the RPC portmapper).

By default, when an HPSS server starts, it will select a random ephemeral port number on which to listen for connections. It registers this port number with the portmapper daemon running on that host. When a client application wishes to communicate with the server, it will contact the portmapper to find out which port the server is listening on. It will then connect to the server on that port. The Core Servers and Gatekeepers can be configured to select a listening port from a small range specified by an HPSS environment variable. The client will still have to contact the portmapper to determine which of these ports the server is listening on, but since they are listening on a predetermined fixed range of ports, the firewall can be configured to allow connection through to these ports.

The port range is specified using the HPSS_LCG_SERVER_RPC_PORT_RANGE environment variable specified in the `/var/hpss/etc/env.conf` file on your Core Server machines with the following format:

```
HPSS_LCG_SERVER_RPC_PORT_RANGE=LOW-HIGH
```

where LOW and HIGH are integers that specify the port range. LOW and HIGH are included in the range. For example, if you specify:

```
HPSS_LCG_SERVER_RPC_PORT_RANGE=29913-29915
```

then any Core Server or Gatekeeper that runs on that machine will always use one of the ports: 29913, 29914, or 29915. If no ports in the range are available at the time the server starts it will abort with an error message logged to the SSM *Alarms and Events* window. Therefore, it is important to configure a wide enough range so that each of these servers can be allocated a port. For example, if you have two Core Server machines, A and B with the following configuration:

Machine A:

```
Core Server 1
```

```
Gatekeeper 1
```

Machine B:

```
Core Server 2
```

```
Gatekeeper 2
```

You will have to configure Machine A with a range of at least 3 ports and Machine B with at least 2 ports. Note that the range of ports for the two machines can overlap (or be the same), since it is the combination of the machine interface name and the port number that must be unique.

It is sometimes convenient to configure the port range to larger than the minimum number. When you stop a server, or it exits, the machine operating system will keep that port in a TIME-WAIT state for several minutes before allowing reuse of the port. Configuring an extra one or two ports in the range will allow you to stop and restart a server without having to wait for the port to time out.

7.7.1. Web SSM and SSM API

Consider what sort of access users should have to the web-based SSM and SSM API when configuring core server firewall rules. The administrator may want to consider restricting access to these interfaces when setting up the firewall. Sites will want to select a web service port that is not in the HPSS_LCG_SERVER_RPC_PORT_RANGE.

7.8. Semaphore values

Adjust the semaphore values (do this for the Core Server and Movers). The example that follows is for a system with 64 GB of memory:

1. Determine the amount of system memory.

   ```
   # grep "MemTotal" /proc/meminfo
   MemTotal:      67108864 kB
   ```

   Memory in bytes = 67,108,864 × 1,024 = 68,719,476,736
   Memory in GB = 67,108,864 / 1,024 / 1,024 = 64

2. Calculate the following variables which will be used to set the semaphore settings in `/etc/sysctl.conf`.

   In RHEL 7.8 a 32768 limit was added to the SEMMNI and MSGMNI values. If the calculated value

is above the new limit then 32768 will need to be used.

Table 14. Kernel parameter expressions

Memory_in_Bytes	**MemTotal × 1024**
Memory_in_GB	MemTotal / 1024 / 1024
shmmni	256 × Memory_in_GB
msgmni	1024 × Memory_in_GB
shmmax	Memory_in_Bytes
shmall	2 × Memory_in_Bytes / 4096
sem	4096 2048000 32 <256 × Memory_in_GB>
msgmax	65536
msgmnb	65536

3. Open `/etc/sysctl.conf` in an editor and add a section to the file to include the entries listed below. If the system variable does not appear in the file, then add it.

   Example:

```
# ----------------------------------
# Semaphore/memory values for HPSS/DB2

kernel.shmmni = 16384
kernel.msgmni = 65536

# Controls the maximum shared segment size, in bytes
kernel.shmmax = 68719476736

# Controls the maximum number of shared memory segments, in pages
kernel.shmall = 33554432

# Semaphore setting
kernel.sem = 4096 2048000 32 12032

# Controls the maximum size of a message, in bytes
kernel.msgmnb = 65536

# Controls the default maximum size of a message queue
kernel.msgmax = 65536

# ----------------------------------
```

4. Set randomize_va_space in `/etc/sysctl.conf`. If this setting does not appear, then add it.

```
kernel.randomize_va_space = 0
```



Here is the OSB (Operational Service Bulletin) describing why this setting is turned off:
https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=support:osb:916_db2_aslr_issue_

5. Set the hung task timeout to 900 seconds when setting up a system which will be used as a tape Mover. This will prevent the system from getting hung task kernel messages during long-running tape operations.

```
kernel.hung_task_timeout_secs = 900
```

6. Commit the changes to `sysctl.conf` to the current environment.

```
# sysctl -p
```

7.9. Enable Core Dumps

While HPSS is generally very stable, at times a site may encounter a server crash. Since we can never know when a crash is likely to occur or whether it will be able to be triggered again, core dumps should always be configured on for HPSS Core, Mover, and Client systems so that we can capture information about a crash when it occurs.

HPSS manages its own core dump locations in `/var/hpss/adm/core`, rather than being processed by systemd-coredump. This enables HPSS to capture core files in a known, well-provisioned location, split by server, for ease of identification and collection.

Modify the system core dump location to disable systemd-coredump processing:

```
$ sysctl -w kernel.core_pattern=core
```

Edit the `/usr/lib/sysctl.d/50-coredump.conf` to prevent systemd-coredump from reverting the change:

```
kernel.core_pattern=core
```

You can check for the correct setting by reviewing the ulimit settings for root and other HPSS user profiles. It should be unlimited:

```
$ ulimit -c
unlimited
```

If it is not unlimited, review your operating system configuration materials on enabling core dumps. Generally, the steps involved require setting the core file ulimit. This should be done for

root and all other HPSS user profiles:

```
ulimit -c unlimited
```

another common solution is to change the system-wide core file size soft limit in `/etc/security/limits.conf` like:

```
*               soft    core            unlimited
```

HPSS core files can be very large (gigabytes), based upon various configuration parameters such as the number of concurrent threads and configured I/O limits. We suggest setting the limit to unlimited to avoid a situation where we receive a truncated core file. Truncated core files are not useful in troubleshooting.

Installations should test that core dumps are occurring correctly by sending **kill -ABRT <hpss process id>** to an HPSS process such as `hpss_ls`. This will cause the server to die and attempt to drop a core dump in a subdirectory under `/var/hpss/adm/core`. Verify that a nonzero-sized core dump has appeared and restart the process that was killed.

Chapter 8. HPSS installation and infrastructure configuration

This chapter provides instructions and supporting information for installing the HPSS prerequisite software, extracting the HPSS and DB2 software from the HPSS distribution media, and performing the HPSS infrastructure configuration.

To install and set up an HPSS system, we recommend that the administrator be familiar with UNIX commands and configuration, be familiar with a UNIX text editor, and have some experience with shell scripts.



For information on upgrading from a previous version of HPSS, see instructions provided by HPSS support.

The steps required to install and set up an HPSS system are listed below. Each step is discussed in more detail in the section referenced.

Prepare for installation

Install prerequisite software

Install HPSS with RPMs

Configure HPSS infrastructure

Post-installation procedures

Locate HPSS documentation and set up manual pages

Define HPSS environment variables

Tune DB2

8.1. Prepare for installation

The following sections discuss the steps that need to be taken in order to prepare the system for HPSS installation.

8.1.1. Distribution media

Obtain the HPSS software from HPSS support.

8.1.2. Software installation packages

The HPSS software is provided in the following packages:

- HPSSSource-<*release*>.tar.Z - Contains HPSS source files.

- SSMHelp.<*release*>.tar - Contains the management guidance of this document in HTML format which is used by the SSM's Help menu to display window-specific help topics.
- hpss.<*release*>_doc.tar - Contains all HPSS documentation in PDF format.

8.1.3. Create owner account for HPSS files

The HPSS software must be installed by a **root** user. In addition, a UNIX User ID of *hpss* and Group ID of *hpss* is required for the HPSS installation process to assign the appropriate ownership for the HPSS files. If the *hpss* User ID does not exist, the installation process will create it with the **mkhpss** tool based on default UID and GID values defined by the `~hpss/include/hpss_env_defs.h` file. If alternate IDs are required, this include file should be updated prior to building, or rebuilding, **mkhpss**, or the *hpss* account/group should be created beforehand using standard administrative procedures. Also, note that HPSS will need a number of other user and group IDs for various servers and prerequisite components. These IDs are also predefined in the same include file and can be modified to support a site's particular account policy.



It is very important that the HPSS file permissions be set up properly. If they are not, HPSS may not work properly after it is configured. We recommend that the HPSS file ownerships and permissions set by the installation process be preserved. If they must be changed, care must be taken to ensure they are changed correctly. Refer to Post-installation procedures for more information on the HPSS file ownerships and permissions.

8.1.4. Installation target directory preparation

By default, the HPSS software is installed in the `/opt/hpss` directory. Before installing the HPSS software, make sure that the installation target directory satisfies the following conditions:

- The `/opt/hpss` directory is not being used.
- The disk, where the installation target directory resides, has enough space to hold all the HPSS packages to be installed on this node.



Do not use NFS-mounted directories for installing nor for allocating space for HPSS-related components. Installing on NFS is problematic and the errors can be difficult to diagnose.

8.2. Install prerequisite software

This section provides an overview of how to install the prerequisite software to prepare for the upcoming HPSS configuration. Verify that the correct software versions are obtained as described in the release notes for the version being installed at https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=docs:release_notes.

8.2.1. Install Java

Java is required to compile HPSS software. It is also required to run the SSM Command Line Interface (hpssadm(7)). If the site needs to compile the HPSS code, the Java Software Development

Kit (SDK) must be installed. To run HPSS applications, either the Java SDK or the Java Runtime Environment (JRE) is required.

It is recommended that the Java SDK component be installed on the machines where HPSS code is to be compiled. Also, the JRE component should be installed on each SSM client machine. This is to enable the hpssgui(7) application to run directly on the client machine to maximize hpssgui(7) performance and to eliminate the X traffic on the HPSS server machine.

See Java to obtain the download website. Follow the installation instructions provided on the website to install Java. Be sure that all Java's prerequisite requirements are met before installing the product.

8.2.2. Install Jansson

HPSS uses the Jansson library to handle JSON inputs and outputs. Jansson is included with RHEL distributions, but make sure that it matches or exceeds the level in the *Release Notes*.

8.2.3. Install TI-RPC

HPSS uses the transport-independent RPC runtime (TI-RPC). This is included with RHEL distributions. Ensure that the level installed matches or exceeds the level in the *Release Notes*.

8.2.4. Install Ncurses

The ncurses library is required to run the **mkhpss** tool. Ncurses is included with RHEL distributions. Follow the instructions provided with the installation files to install ncurses.

8.2.5. Install MIT Kerberos



The capability to use MIT Kerberos authentication is provided in HPSS; however, IBM Service Agreements for HPSS do not provide support for defects in MIT Kerberos. Kerberos maintenance and support must be site-provided.

For Linux, Kerberos is normally included in the operating system and does not need to be installed.

8.2.6. Install LDAP (if using LDAP authorization)



LDAP authorization is not supported by IBM Service Agreements. The following information is provided for sites planning to use LDAP authorization with HPSS as a site-supported feature. If UNIX authorization will be used, this product is not required.

LDAP authorization is supported on all platforms.

The OpenLDAP client API is the only supported API in HPSS.



If planning to use the LDAP option, make sure you get the LDAP RPM set, or use the LDAP authentication provided by PAM via HPSS UNIX authentication. For

> *OpenLDAP, set SASL_INSTALL, SSL_INSTALL, and LDAP_INSTALL_PATH, to the locations of their respective packages, and set LDAP_TYPE to OPENLDAP.*

8.2.7. Install DB2 and set up permanent license

DB2 is provided as part of HPSS, though it is a separate installation. The DB2 installation image and related license file can be found on the DB2 Installation CD or image. It can also be obtained by contacting HPSS support. Untar the DB2 installation package (if necessary) onto a local file system.

Use the below commands to install DB2. We will use VX.Y to denote the version of Db2 being installed in this procedure.

```
% su -
% cd <location of DB2 install files>
% cd server
Note: Make sure all of db2 prerequisites have been installed
% ./db2prerequisitecheck
% ./db2_install
```

To create a permanent DB2 license, issue the following commands:

```
# cd /opt/ibm/db2/VX.Y/adm

# ./db2licm -a <path name to the DB2 generic license file>
```

Refer to the *DB2 Command Reference* document for more information on how to use the **db2licm** utility to manage the DB2 license.

8.2.8. Install HPSS License

HPSS requires license files to run with all features enabled. The license files will be provided to you by HPSS Support for each HPSS instance. You should expect to periodically get a new license from HPSS Support.

Copy the license files into HPSS_PATH_ETC (e.g. /var/hpss/etc). The license files are:

- license.json (readable json file with the license settings)
- license.sig (encoded license file)

Both files are required to be installed, readable by the user account with which the HPSS Core Server runs, and not altered. These files should exist prior to starting the Core Server, since otherwise the Core Server may be degraded. See [invalid-license] for more information.

The HPSS Core Server will check for the license file and validate it periodically.



HPSS does not require an internet connection to validate its license file.

8.2.9. Install OIDC Prerequisites

Sites must have their own Open ID Connect (OIDC) provider. HPSS does not provide a production-ready OIDC provider. See [configuring-sample-keycloak] on setting up the sample Keycloak.

The SSM API will need to be configured to reference the site's OIDC provider. This process is described in Configuring nginx for Reverse Proxying.

8.2.10. Install Web Service Prerequisites

HPSS Web Services (SSMAPI, WebSSM) should be run on the HPSS core server system. For ease of deployment, nginx should also be enabled on the same server. However, nginx can be set up to proxy HPSS web services with a simple modification of the template file.

The HPSS SSM API contains two Python packages which describes the Python modules that must be installed: smclient and ssmapi. The HPSS Web SSM UI also provides its own Python module, webssm.

The required Python modules will be installed automatically when the HPSS web service RPM is installed if the packages are available in the local pip repository. If the system does not have access to a pip repository, then the required packages listed in the ssmapi and smclient package METADATA file should be installed or made available to the local pip repository.

8.3. Install HPSS with RPMs

This section details the steps necessary to install HPSS using RPMs.

HPSS RPMs can be obtained either from the HPSS Admin Wiki or from the HPSS Collaboration website for registered HPSS customer sites. RPMs are the preferred method of installing HPSS software.

1. https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=start
2. https://hpss-collaboration.org

Installation of HPSS with RPMs is simple. RPM installation is done with:

```
rpm -ivh <rpm-name>
```

There are ten main RPM packages for HPSS:

Table 15. RPM packages

Package	Description
hpss-lib	This is a dependency for all the other RPMs except for hpss-doc. It contains a prerequisite set of binaries required to run HPSS such as shared object files and a static library. Also, it contains source include files and make files for running tools such as hpss_db2_bindall. Finally, it includes files for writing Python applications that interface with the HPSS RPC.
hpss-lib-devel	This contains include files required for compiling an HPSS API program from source.
hpss-core	This contains executable files and dependent source files required for setting up and running an HPSS Core server Machine. Installing it also sets up a link to the default IBM DB2 installation.
hpss-webssm	This provides functionality for accessing a browser-based GUI to view status information in HPSS. Also, it provides a RESTful SSM API for writing Python applications that interface with HPSS.
hpss-mvr	This contains binary executable files and source configuration files required for running an HPSS Mover.
hpss-clnt	This contains binary executable files for running client applications such as pftp.
hpss-clnt-devel	This contains source include files for developing HPSS client applications.
hpss-clnt-devel-mvrprot	This contains confidential header files for developing applications that use the HPSS Mover protocol. *This RPM is only available at special request.*
hpss-pvr	This contains executable files and dependant source files required for running a remote PVR. Installing it also sets up a link to the default IBM DB2 installation.
hpss-doc	This contains HPSS documentation files in the form of HTML and PDF files for the error manual, admin guide, programmer's reference, and the user's guide. It also contains the files required to access help menus in the HPSS GUI.

For a typical installation, you must install hpss-lib, which is a dependency for most of the other RPMs. You would then install hpss-core if you needed core binaries, hpss-mvr for a Mover, and hpss-clnt for client functionality. After hpss-lib is installed, you can mix and match functionality between hpss-core, hpss-mvr, and hpss-clnt.



The hpss-pvr has some binaries which conflict with hpss-core, so these two packages should not be installed on the same machine. However, you can install the hpss-pvr, hpss-mvr, and hpss-clnt packages on the same machine.



The full name of an RPM also contains the HPSS version as well as the machine

architecture and operating system version built for. If installing on RHEL, install the RPMs containing the appropriate operating system version ("el8" for RHEL8 and "el9" for RHEL9) and the appropriate machine architecture (ppc64le for PPC Little Endian Linux or x86_64 for x86_64 Linux).

8.4. Install HPSS

This section details the steps necessary to install HPSS on the root, Movers, and other client nodes.

8.4.1. On core

To install HPSS on a Core Server machine, the RPMs hpss-lib and hpss-core should be installed, for example:

```
# rpm -iv hpss-lib-X.Y-0.el8.x86_64.rpm hpss-core-X.Y-0.el8.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y-0.el8
Files for package hpss-lib installed under
/opt/hpss
hpss-core-X.Y-0.el8
ln -sf /opt/ibm/db2/default /opt/hpss/db2
Files for package hpss-core installed under
/opt/hpss
```

Note that the install sets up a link file under the HPSS installation directory to point to the default DB2 link.

Note that if you will be installing HPSS on a different machine as a Mover, client, or remote PVR, you should create a tar file of the `/var/hpss/etc` directory, for example:

```
# tar -cvf /tmp/var_hpss_etc.tar /var/hpss/etc
```

After doing this you should create a `/var/hpss/etc` directory on the non-core machine and untar the above tar file to them.

8.4.2. On Mover

To install HPSS on a Mover machine, the RPMs hpss-lib and hpss-mvr should be installed, for example:

```
# rpm -iv hpss-lib-X.Y-0.el8.x86_64.rpm hpss-mvr-X.Y-0.el8.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y-0.el8
Files for package hpss-lib installed under
/opt/hpss
hpss-mvr-X.Y-0.el8
```

```
Files for package hpss-mvr installed under
/opt/hpss
```

8.4.3. On client

To install HPSS on a client machine, the RPMs hpss-lib and hpss-clnt should be installed, for example:

```
# rpm -iv hpss-lib-X.Y-0.el8.x86_64.rpm hpss-clnt-X.Y-0.el8.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y-0.el8
Files for package hpss-lib installed under
/opt/hpss
hpss-clnt-X.Y-0.el8
Files for package hpss-clnt installed under
/opt/hpss
```

8.4.4. On remote PVR

To install HPSS on a remote PVR machine, the RPMs hpss-lib and hpss-pvr should be installed, for example:

```
# rpm -iv hpss-lib-X.Y-0.el8.x86_64.rpm hpss-pvr-X.Y-0.el8.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y-0.el8
Files for package hpss-lib installed under
/opt/hpss
hpss-pvr-X.Y-0.el8
ln -sf /opt/ibm/db2/default /opt/hpss/db2
Files for package hpss-pvr installed under
/opt/hpss
```

8.4.5. Generate and bind the DB2 helper program

When filesets are created or updated, it is sometimes necessary to make entries in both the global and the subsystem database. When updating both of these databases, it is very important that the update be performed atomically. So, to accomplish an atomic update, a DB2 helper program is created. This DB2 helper program is *run* by DB2 whenever it needs to perform an atomic update of the global and subsystem databases when creating or updating a fileset.

To generate this DB2 helper program and bind it to DB2 a script named **hpss_db2_bindall.ksh** is provided.

Whenever the base source tree is rebuilt, perform the following steps to generate and bind the DB2 helper program:

1. Log on as *hpss*.

2. Change directory to `$HPSS_ROOT/bin`.
3. Run the following command:

```
% hpss_db2_bindall.ksh
```

8.4.6. Install SSL certificates for web services

Obtain a SSL certificate signed by a trusted Certificate Authority (CA) such as Let's Encrypt. To install this, you need to place both your certificate file (with a .crt or .pem extension) and your private key anywhere under `/etc/`. For example `/etc/ssl/certs/` for the certificate, and `/etc/ssl/private/` for the private key.



When installing an SSL certificate, it's crucial the certificate common name (CN) or Subject Alternative Name (SAN) matches the domain name you will use for every HPSS web service. For example, if your website is `example.com`, your SSL certificate must be issued for `example.com`, and vise versa for subdomains such as `subdomain.example.com`. Mismatched domain names will cause security warnings even if the certificate is signed by a trusted CA.

A site should have a plan or have an automated process to ensure that the SSL certificates are renewed before they expire such as a cron job or shell script.

It should also be noted that self-signed certificates should only be used in specific cases, such as internal testing as they are not trusted by browsers and will trigger security warnings. For production environments, it is strongly recommended that sites obtain or use a certificate from a trusted CA to ensure proper security and trust. If a site is considering using a self-signed certificate, please reach out with your HPSS support representative to discuss your plans and determine the best and most secure approach.

8.4.7. Update default DB2 link

HPSS can support multiple versions of DB2; see release notes for specific versions.

HPSS uses a default link for the DB2 installation path. This link is `/opt/ibm/db2/default` and should point to the DB2 installation path. Another link, `$HPSS_ROOT/db2`, should point to `/opt/ibm/db2/default` by default. The link `$HPSS_ROOT/db2` can be modified if multiple HPSS installations with different DB2 requirements must coexist on the same machine.

When HPSS is installed with RPMs, the `$HPSS_ROOT/db2` — typically `/opt/hpss/db2` — link is set to point to `/opt/ibm/db2/default`. However, the administrator must ensure that `/opt/ibm/db2/default` points to the correct DB2 installation. DB2 is typically installed in `/opt/ibm/db2/VX.Y`; for example, `/opt/ibm/db2/V10.5`.

When updating or changing the DB2 version used, the administrator should update `/opt/ibm/db2/default` to point to the correct DB2 installation. In the case of a system with multiple installations with different DB2 requirements, the `$BUILD_TOP_ROOT/db2` link for the installation, which will not use the default DB2 version, should be changed to point to the correct DB2 version.

8.5. Configure OIDC Provider

OIDC providers enable web service workflows like limited time access tokens, two factor authentication, and more. Sites must have their own OIDC provider configured and operational before configuring the SSM API. Optionally, depending on site requirements, HPSS provides a **configure_keycloak** script, documented in Configuring Keycloak, that can help assist in the initial setup with a sample Keycloak image.

The site's OIDC provider will need to be populated with the HPSS user accounts that the site plans to give access to HPSS web services. HPSS administrator or HPSS operator users should have access to HPSS web services as OIDC users (with the same usernames). HPSS web services will use the username to determine the level of access the incoming OIDC user should have.

8.5.1. Configuring Keycloak

HPSS provides a sample Keycloak container image for testing and prototyping HPSS OIDC integration. Keycloak is not bundled as a production component.

Use the `configure_keycloak` script to install and configure a sample Keycloak OIDC provider.

The script will:

- Create an initial Keycloak administrator account
- Prompt for the administrator password
- Use the provided sample Podman compose file to deploy a Keycloak container named `hpss_keycloak`



In a production environment, Keycloak must run behind an edge-terminated reverse proxy. This sample setup is not production-ready.

After the script is complete, the Keycloak container `hpss_keycloak` can be started with:

```
# systemctl start hpss_keycloak
```

The Keycloak Admin Console is available only on localhost of the machine where Keycloak is installed and running:

http://localhost:8080/auth/admin/master/console/



The Admin Console is intentionally restricted to localhost without HTTPS. Exposing admin endpoints behind a reverse proxy unnecessarily increases attack surface.

To the update the Keycloak administrator password:

1. Open the Admin Console
2. Click **Users**

3. Select the admin user
4. Open the **Credentials** tab
5. Click **Reset password**
6. Enter the new password
7. Click **Save** to confirm

To view or change the Client Secret within Keycloak:

- Select the sample "hpss" realm from the drop down in the top-left corner
- Click "Clients" on the left-hand side
- Select "hpss" from the "Clients List"
- Select the "Credentials" tab
- From this window you can view a current "Client Secret" or regenerate a new "Client Secret"

To set a valid redirect URI(s):

- Inside of the sample "hpss" realm, click "Clients" on the left-hand side
- Click on the "hpss" client
- On the "Settings" tab, enter the callback URIs (see Quick startup of the SSM API for getting the callback URI for the SSM API)



An invalid callback URI in Keycloak will result in a **redirect_uri** error.

To change the Client Secret within Keycloak:

- Select the sample "hpss" realm from the drop down in the top-left corner
- Click "Clients" on the left-hand side
- Select "hpss" from the "Clients list"
- Select the "Credentials" tab
- From this window you can view a current "Client Secret" or regenerate a new "Client Secret"

8.5.2. Configuring Your OIDC Provider for HPSS



For production environments, it is critical that the OIDC service runs using https or behind an edge-terminated reverse proxy and the web server serving it has valid certificates installed.

Configuration steps will be specific to your OIDC provider. Consult your OIDC provider's documentation for detailed setup instructions. The following provides general guidance:

- Ensure your OIDC provider is properly configured and operational
- Create a client/application registration in your OIDC provider for HPSS
- Obtain the client ID and client secret from your OIDC provider

- Configure the SSM API to use the correct OIDC variables when running configure_ssmapi. See Quick startup of the SSM API for details.
- Add HPSS users to your OIDC provider with the same usernames as their HPSS SSM accounts

Users can be added through your OIDC provider's administrator console. Consult your OIDC provider's documentation or work with your local identity management team to add the appropriate users.

8.6. Configure HPSS infrastructure

The information in this section will describe how to use the **mkhpss** utility to configure HPSS and DB2. The HPSS installation and infrastructure configuration must be performed on the following HPSS machines:

- Root subsystem
- Subsystems created via "Install Subsystem" tab in **mkhpss**

Although the installation and configuration steps to be performed are similar, each step will be processed differently depending on the machine type. Therefore, it is very important that the correct submenu (Root Subsystem, Secondary Subsystem) is selected to perform the configuration. The following sections describe the procedure to be performed on each machine.

8.6.1. Navigating and general mkhpss behavior

Prior to using **mkhpss**, a quick word on navigation and general behaviors.

There are four window panels in **mkhpss**. At the bottom left, the keyboard shortcut window displays the **mkhpss** keyboard shortcuts. This is a static window that cannot be selected. At the top left, the menu window shows the various configuration screens that can be selected. At the top right, the content window shows the currently selected menu selection. At the bottom right, the message window displays information about configuration actions and results.

A box in the bottom left displays the keyboard shortcuts. They are:

Tab	Cycles to the next field. The content window often has multiple fields, and the Tab key will cycle through each field in the window.
Shift+Tab	Same as Tab, but cycles in reverse.
F4	Exit. This exits **mkhpss**.
F6	Toggle Active Window. This will change the focus of the active window, allowing scrolling or selection in that window. The keyboard shortcut window is never active.

Users can also navigate between panels and fields using the mouse.

When the *Configure* button is pressed, **mkhpss** will run a series of configuration scripts, whose output will go in the message window. During this time, inputs to **mkhpss** will be ignored.

The **mkhpss** utility logs the messages window to `/var/log/mkhpss.log` and appends to this file each time it runs. You must have write access to this file in order to run **mkhpss**. Each time scripts are run via *Configure*, a timestamp and duration in seconds will be appended to the file.

8.6.2. Configure HPSS - root subsystem machine

The root subsystem machine is the machine upon which the root Core Server executes. Perform the following steps on that (root subsystem) machine:

Pre-installation configuration

Configure HPSS security services

Configure DB2 services

Configure other services

Configure other services

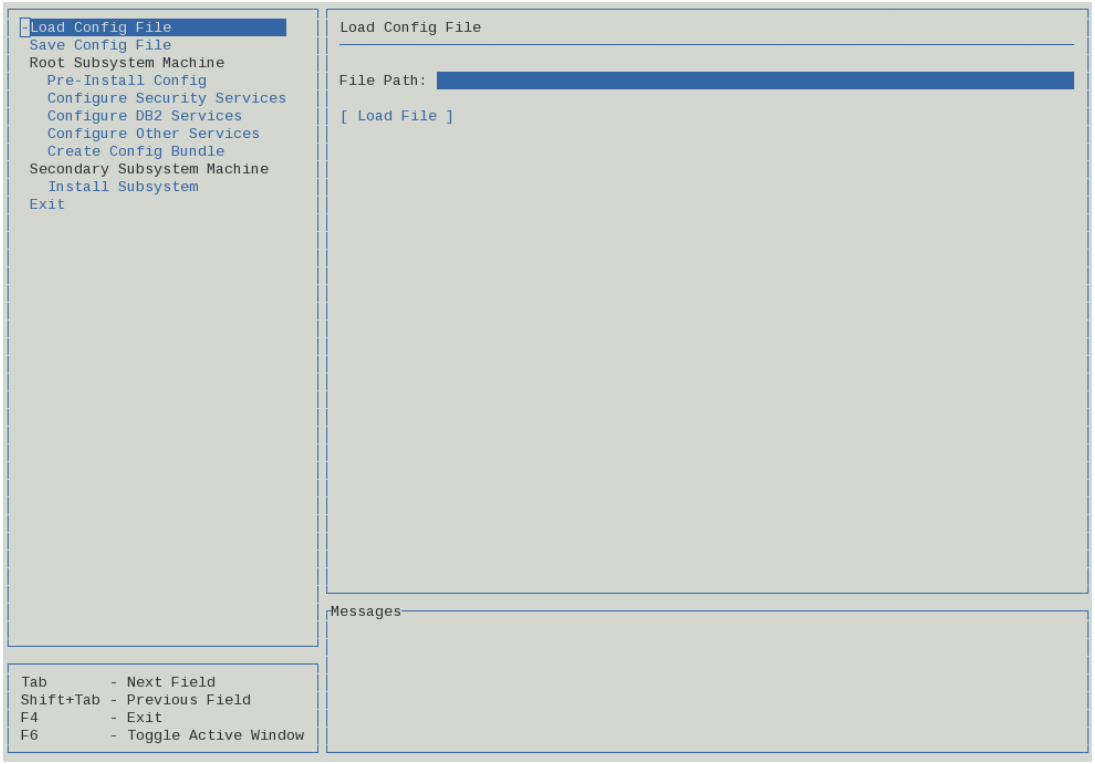
Create configuration bundle

The following sections describe the procedures to perform these steps.

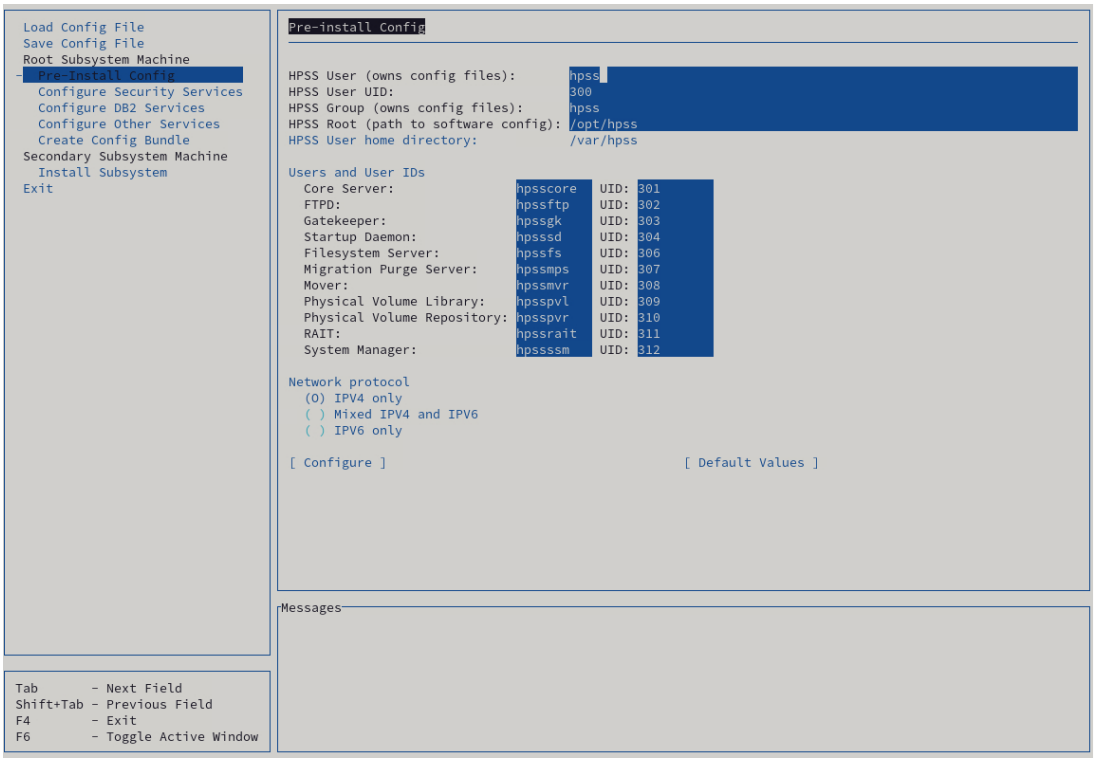
8.6.2.1. Pre-installation configuration

The **mkhpss** utility is used to set up HPSS user accounts, authentication, authorization, metadata resources, and initial system management configuration. It must be run prior to configuring HPSS software services and storage, but after other HPSS prereqs such as DB2 are installed.

1. The RPM install process will place **mkhpss** in the HPSS RPM `bin` directory, referred to as `$HPSS_ROOT/bin`. For example, since HPSS is installed at `/opt/hpss/`, the **mkhpss** tool should be in the `/opt/hpss/bin` directory. Invoke the **mkhpss** utility and you should see the following screen:



2. From the "Root Subsystem Machine" submenu, select the *Pre-Install Config* option in the content panel. **mkhpss** will display the following screen:



3. Verify that the default values are correct for the given installation and modify if necessary. Make sure that the server account UIDS in `/etc/passwd` or the local `/var/hpss/etc/passwd` (depending on authentication settings), either exist as the same UID or do not exist. Select the *Configure* button to perform the pre-installation setup. This will run a set of scripts to verify/create the *hpss* account and group, set up the `/var/hpss` directory with the required subdirectories and initialize the HPSS environment file, `env.conf` in `/var/hpss/etc`.
4. Check for any errors in the "Messages" window. When the execution is complete, the following

line will display:

```
run command finished: mkhpss_run_preinstall
```

This will be followed by a timestamp and the duration of the command.

8.6.2.2. Configure HPSS security services

The following sections describe the procedure to configure common security combinations using the **mkhpss** utility.

Configure UNIX authentication and UNIX authorization

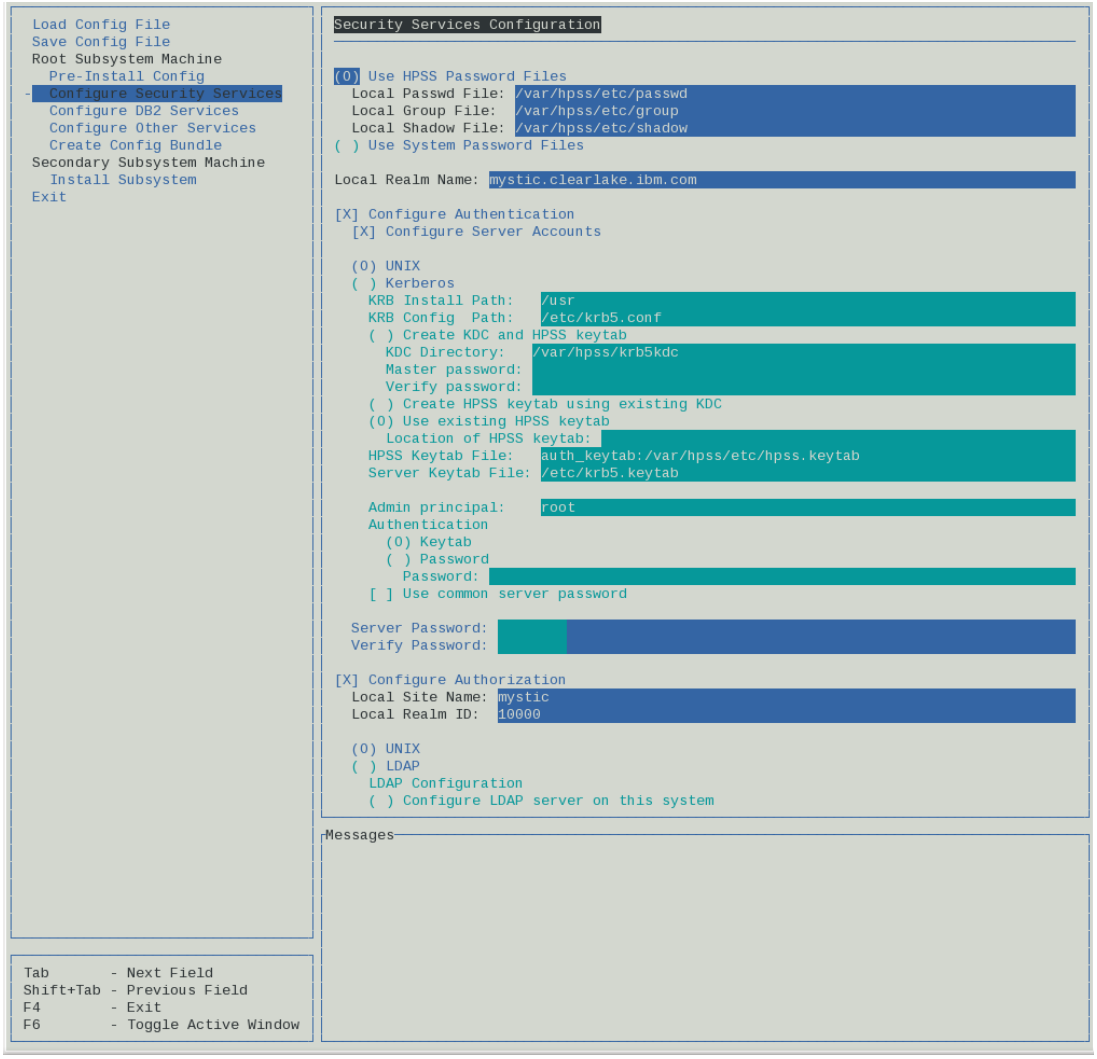
From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Security Services" option. Perform the following steps using the content panel:

1. Select either local or system password files based on how the site wants accounts and passwords to be managed. By default, "Use HPSS Password Files" is selected with the HPSS configuration files set locally (`/var/hpss/etc/passwd`, `/var/hpss/etc/group`, `/var/hpss/etc/shadow`, and `/var/hpss/etc/kt.shadow`) to administer the authentication and authorization services. As an option, "Use System Password Files" can be used instead (`/etc/passwd`, `/etc/group`, `/etc/shadow`, and `/var/hpss/etc/kt.shadow`).

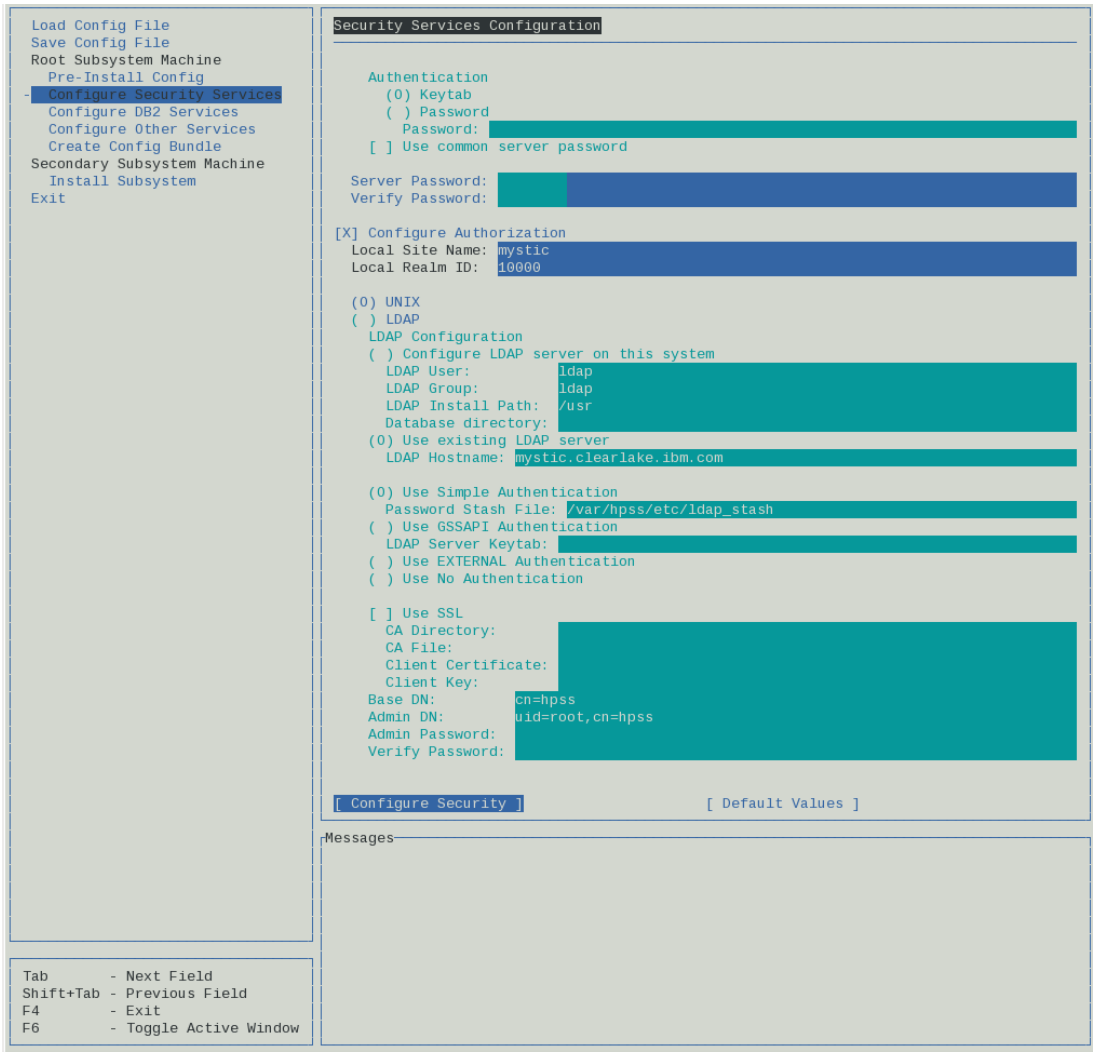
   Local Realm Name

   By convention, this value is usually set to the fully qualified domain name of the local host.

2. By default, the "Configure Authentication" check box is selected.
3. By default, the "Configure Server Accounts" and UNIX authentication check boxes are selected to create UNIX accounts for HPSS servers.
4. The fields to be set are as displayed on the screen shown:



5. Using the keyboard, scroll down through the content panel until the "Configure Authorization" fields are visible. It should look like the following:



6. By default, the "Configure Authorization" check box is selected.
7. Review and modify (if necessary) the active fields:

   Local Site Name

   The value is usually set to the fully qualified host name which can be determined using the **hostname** and **domainname** commands.

   Local Realm ID

   The field is set to a unique ID number for each site. Ask HPSS support for your site's value.

8. By default, "Authorization Service" is set to "UNIX".
9. Click on the "Configure Security" button at the bottom of the screen to perform the specified security configuration.
10. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

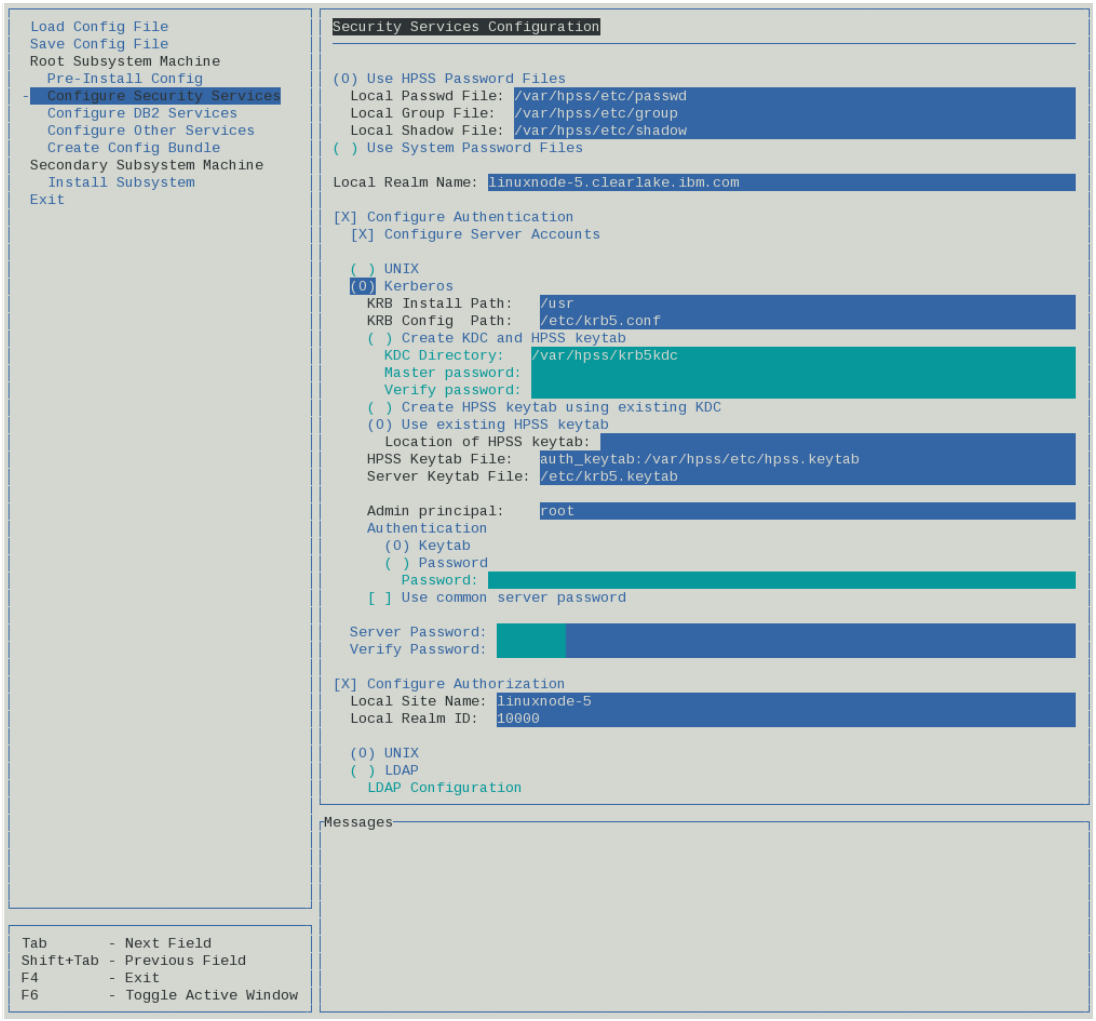
```
run command finished: mkhpss_run_core_sec_config
```

This will be followed by a timestamp and the duration of the command. This step sets up the hpss_unix_login service (hpss_login.service).

Configure Kerberos authentication and UNIX authorization

From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Security Services" option. Perform the following steps using . By default, the "Configure Authentication" check box is set.

1. Keep the "Configure Server Accounts" check box selected, but change the default setting from UNIX to Kerberos to create accounts for HPSS servers.
2. Select either local or system password files based on how the site wants accounts and passwords to be managed.
3. The fields to be configured are displayed on the screen shown:



4. Review and modify (if necessary) the following authentication fields:

Use HPSS Password Files

By default, selected with the HPSS configuration files set locally (`/var/hpss/etc/passwd`, `/var/hpss/etc/group`, `/var/hpss/etc/shadow`, and `/var/hpss/etc/kt.shadow`) to administer the authentication and authorization services. As an option, "Use System Password Files" can be used instead (`/etc/passwd`, `/etc/group`, and `/etc/shadow`). However, `/var/hpss/etc/kt.shadow` will always be needed for either flavor of UNIX. Other HPSS utilities are available to administer these HPSS configuration files. See the Security mechanisms section for more information.

Local Realm Name

By convention, this value is usually set to the Fully Qualified Domain Name of the local host. If you are not using **mkhpss** to create the Kerberos Key Distribution Center (*Create KDC and HPSS keytab* is not checked), then the Local Realm Name must be set to the name of the Kerberos Realm that will be used by HPSS. If *Create KDC and HPSS keytab* is checked, then the Kerberos authentication system will be created using the *Local Realm Name* as the Kerberos Realm Name.

5. By default, the "Configure Authentication" check box is set.
6. Keep the "Configure Server Accounts" check box selected, but change the default setting from UNIX to Kerberos to create accounts for HPSS servers.

Kerberos Install Path

The path to where Kerberos is installed. The default directory for Red Hat Linux is `/usr`.

Kerberos Config Path

The path to the Kerberos config file. The default path is set to `/etc/krb5.conf`.

Create KDC and HPSS keytab

If desired, **mkhpss** may be used to configure a Kerberos Key Distribution Center (KDC) on the Root Subsystem Machine. This includes creating a new Kerberos Principal database, establishing the Kerberos Realm and execution environment, as well as configuring HPSS startup and shutdown scripts to start and stop the Kerberos services. To create a KDC on the Root Subsystem Machine, select the "Create KDC and HPSS keytab" check box. If your site has an established Kerberos infrastructure which will be used for HPSS, or if you wish to create a KDC housed on a machine other than the Root Subsystem Machine, be sure that the "Create KDC and HPSS keytab" check box is not selected.

KDC Directory

The pathname of the KDC directory. This value is used when the "Create KDC and HPSS keytab" check box is checked. Default setting is `/var/hpss/krb5kdc`.

Master Password

The Kerberos administration password. This option is only available when *Create KDC and HPSS keytab* is checked.

Verify Password

Re-enter the Kerberos administration password. This option is only available when *Create KDC and HPSS keytab* is checked.



Be sure to remember this password to be able to administer the Kerberos environment later.

Create HPSS keytab using existing KDC

Creates an HPSS keytab using the existing KDC environment.

Use existing HPSS keytab

Use an existing HPSS keytab and an existing KDC environment. Enter the location of HPSS keytab.

HPSS Keytab File

Path to HPSS keytab file. Default setting is `auth_keytab:/var/hpss/etc/hpss.keytab`.

Server Keytab File

Path to server keytab file. Default is set to `/etc/krb5/keytab`.

Admin principal

The userid to administer the Kerberos environment.

Authentication

There are two supported options: Keytab File or Password. The Keytab File option allows HPSS servers or utilities to read a keytab file to authenticate. The Password option requires a password to be supplied each time an HPSS server or utility is invoked.

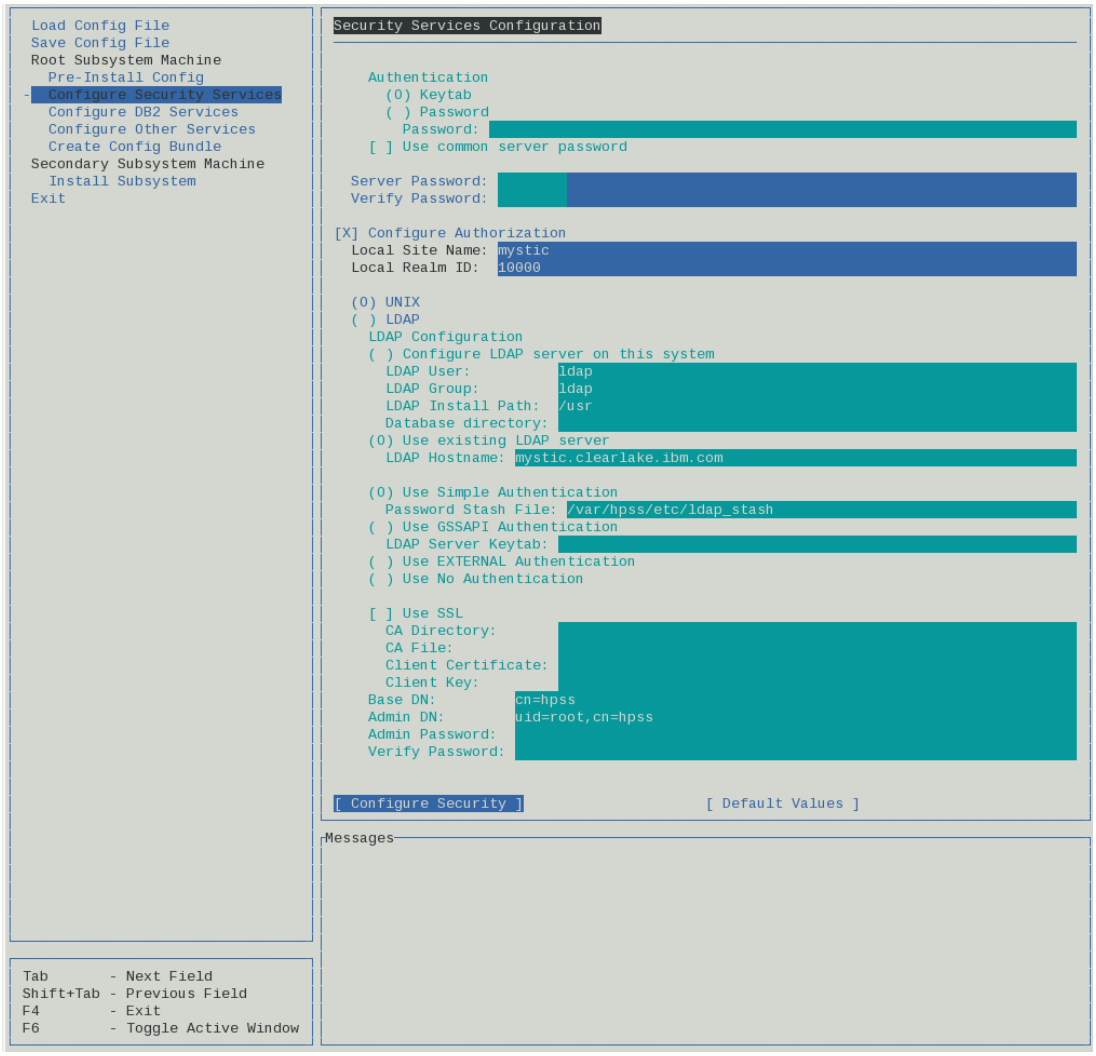
Keytab File

The pathname of the keytab file to be created if the Authentication Type is set to "Keytab File". This field is not enterable when the Authentication Type field is specified as Password.

Password

The password used to authenticate the caller when the HPSS server or utility is invoked. This field is not enterable when the Authentication Type field is specified as "Keytab File".

7. Using the keyboard, scroll down the content panel display until the "Authorization Service" information is seen. It should look like the following:



8. By default, the "Configure Authorization" check box is selected.
9. Review and modify (if necessary) the following authorization fields:

   Local Site Name

   The value is usually set to the fully qualified host name of the local host, which can be determined using the *hostname* and *domainname* commands.

   Local Realm ID

   The field is set to a unique ID number for each site. Ask HPSS support for your site's unique ID.

10. Keep the default setting for "Authorization Service" as "Unix".
11. Click on the "Configure Security Services" button at the bottom of the screen to perform the specified security configuration.
12. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_sec_config
```

This will be followed by a timestamp and the duration of the command.

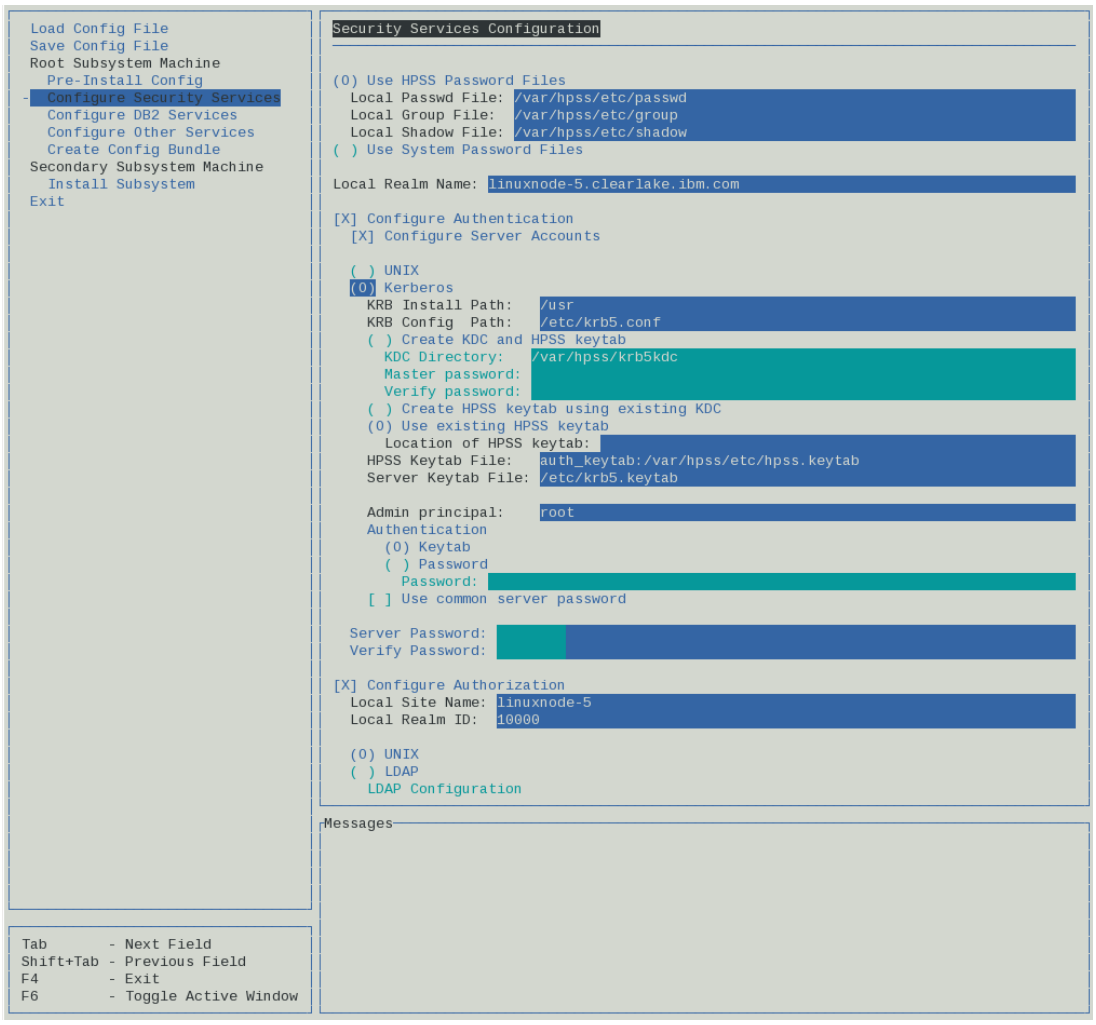
Configure Kerberos authentication and LDAP authorization



UNIX authentication is not supported when LDAP authorization is selected.

From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Security Services" icon. Perform the following steps using the content panel:

1. Select either local or system password files based on how the site wants accounts and passwords to be managed.
2. Select the "Configure Server Accounts" check box to create accounts for HPSS servers. Enter the password to use when creating the server accounts.
3. Select the "Configure Authentication" check box. Set the "Authentication Service" to "Kerberos".
4. The fields to be configured are as displayed on the screen shown:



5. Review and modify (if necessary) the following authentication fields:

Use HPSS Password Files

By default, the system's configuration files (`/etc/passwd`, `/etc/group`, and `/etc/shadow`) are used to administer the authentication and authorization services. As an option, HPSS configuration files can be used instead. These files are created by **mkhpss** as part of this configuration step. Other HPSS utilities are available to administer these HPSS configuration files. See the Security mechanisms section for more information. To use the HPSS

configuration files, select the "Use HPSS Password Files" check box.

Local Realm Name

By convention, this value is usually set to the "Local Site Name" in upper-case letters. If you are not using **mkhpss** to create the Kerberos Key Distribution Center (*Create KDC and HPSS keytab* is not checked), then the Local Realm Name must be set to the name of the Kerberos Realm that will be used by HPSS. If *Create KDC and HPSS keytab* is checked, then the Kerberos authentication system will be created using the *Local Realm Name* as the Kerberos Realm Name.

Kerberos Install Path

The path to where Kerberos is installed. The default directory for RHEL is `/usr`.

Kerberos Config Path

The path to the Kerberos config file. Default path is `/etc/krb5.conf`.

Create KDC and HPSS keytab

If desired, **mkhpss** may be used to configure a Kerberos Key Distribution Center (KDC) on the Root Subsystem Machine. This includes creating a new Kerberos Principal database, establishing the Kerberos Realm and execution environment, as well as configuring HPSS startup and shutdown scripts to start and stop the Kerberos services. To create a KDC on the Root Subsystem Machine, select the *Create KDC and HPSS keytab* check box. If your site has an established Kerberos infrastructure which will be used for HPSS, or if you wish to create a KDC housed on a machine other than the Root Subsystem Machine, be sure that the *Create KDC and HPSS keytab* check box is not selected.

KDC Directory

The pathname of the KDC directory. This value is used when the "Create KDC and HPSS keytab" check box is checked. Default path is `/var/hpss/krb5kdc`.

Master Password

The Kerberos administration password. This option is only available when "Create KDC and HPSS keytab" is checked.



Be sure to remember this password to be able to administer the Kerberos environment later.

Verify Password

Re-enter the Kerberos administration password. This option is only available when *Create KDC and HPSS keytab* is checked.

Create HPSS keytab using existing KDC

Creates an HPSS keytab using the existing KDC environment.

Use existing HPSS keytab

Use an existing HPSS keytab and an existing KDC environment. Enter the location.

HPSS Keytab File

Path to HPSS keytab file. Default setting is `auth_keytab:/var/hpss/etc/hpss.keytab`.

Server Keytab File

Path to server keytab file. Default setting is `/etc/krb5/keytab`.

Admin principal

The userid to administer the Kerberos environment.

Authentication

There are two supported options: Keytab File or Password. The Keytab File option allows HPSS servers or utilities to read a keytab file to authenticate. The Password option requires a password to be supplied each time an HPSS server or utility is invoked.

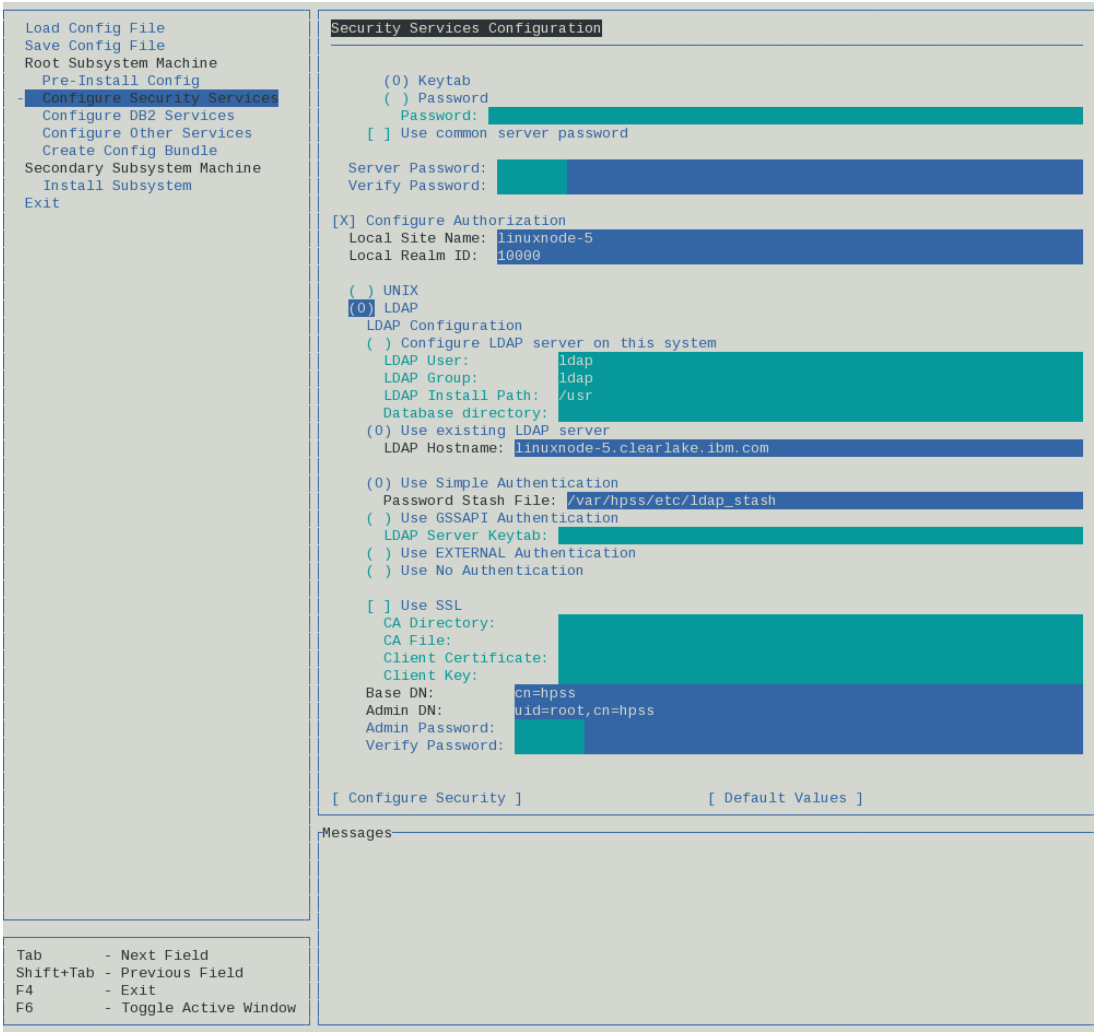
Keytab File

The pathname of the keytab file to be created if the Authentication Type is set to "Keytab File". This field is not enterable when the Authentication Type field is specified as Password.

Password

The password used to authenticate the caller when the HPSS server or utility is invoked. This field is not enterable when the Authentication Type field is specified as "Keytab File".

6. Using the keyboard, scroll down the content panel display until the "Authorization Service" information is seen. It should look like the following:



7. By default, the "Configure Authorization" check box is selected.
8. Review and modify (if necessary) the following authorization fields:

   Local Site Name

   The value is usually set to the fully qualified host name of the local host, which can be determined using the **hostname** and **domainname** commands.

   Local Realm ID

   The field is set to a unique ID number for each site. Ask HPSS support for your site's unique ID.

9. Set the "Authorization Service" to "LDAP".

   Configure LDAP server in this system

   The flag is set to create an LDAP instance locally on this host machine. Tivoli Directory Server cannot run on the same host machine due to the incompatibility with DB2, and there is no support for setting up a database in OpenLDAP. Leave this box unchecked and set up the chosen database instance before configuring.

   LDAP User

   LDAP user name using the local UNIX UID that owns the LDAP server and database.

LDAP Group

LDAP Group name using the local UNIX GID that owns the LDAP server and database.

LDAP Install Path

The path is the installation prefix for OpenLDAP and its utilities. This is where it looks for ldapsearch, ldapadd, and other utilities.

Database directory

Location where the new database will be created.

Use existing LDAP server

Select this if you are using an existing database. This will deselect "Configure LDAP server on this system". The LDAP Hostname is the system that will be configured for all the servers to contact for the LDAP queries. Default will be the Fully Qualified Domain Name of the current host.

Authentication type (if one is used)

Simple, GSSAPI, EXTERNAL

Use Simple Authentication

Authentication via client DN and password.

Password Stash File

Path to the file where the username DN and password are stored for clients.

Use GSSAPI Authentication

Kerberos based and will need the LDAP Server Keytab for client use. Bind DN is derived from the authenticated user name. It takes the form "uid=principal@realm,cn=gssapi,cn=auth".

Use EXTERNAL Authentication

Typically through SASL, the bind DN is derived from the authentication that is performed externally. If connecting through a UNIX socket, it takes the form "gidNumber=<UNIX GID of connecting client>+uidNumber=<UID of client>,cn=peercred,cn=external,cn=auth". If connecting through SSL, the bind DN is the subject of the client's certificate if presented, or anonymous if no certificate is presented.

Use No Authentication

No authentication; client is anonymous. Anonymous access is not recommended.

Use SSL

Optionally, LDAP can be configured with SSL.

CA Directory

Path to certificate authority directory.

CA File

Certificate authority file.

Client Certificate

Path to client certificate file.

Client Key

Path to client key file.

Base DN (Distinguished Name)

Refers to the LDAP base distinguished name. Default value is cn=hpss.

Admin DN (Distinguished Name)

The administrator name that is allowed to add, update, and remove entries in LDAP. Default values are uid=root, cn=hpss.

Administrator Password

The password used by the administrator to manage entries in LDAP.

Verify Password

Repeat of the LDAP administrator password entered to verify it was entered correctly.

10. Click the "Configure Security" button at the bottom of the screen to perform the specified security configuration.
11. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

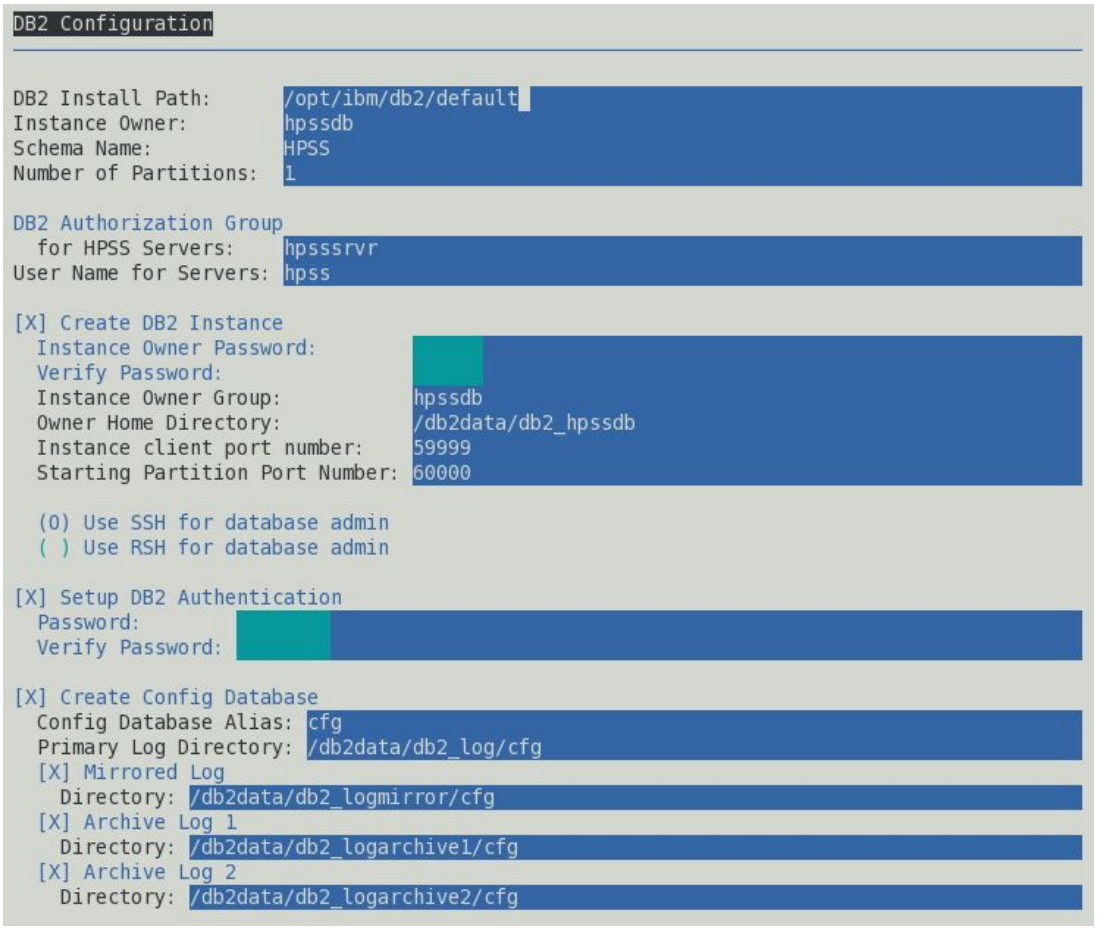
```
run command finished: mkhpss_run_core_sec_config
```

This will be followed by a timestamp and the duration of the command.

8.6.2.3. Configure DB2 services

To configure DB2 databases to manage HPSS metadata, perform the following steps:

1. From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure DB2 Services" option. The following window will be shown:



2. Review and modify (if necessary) the following fields:

DB2 Install Path

The path where DB2 executables are installed. Default path is set to `/opt/hpss/db2`.

Instance Owner

The name of the DB2 instance owner. Normally set to *hpssdb*.

Schema Name

The name of the DB2 schema containing the HPSS metadata table. Normally set to *hpss*.

Number of Partitions

Determines the number of partitions the instance should have. The default setting is "1". Consult with the HPSS Systems Engineering and HPSS DB2 SME teams if your site is considered for number of partition greater than one. See HPSS Planning HPSS/DB2 File Systems chapter.

DB Authorization Group for HPSS Servers

The UNIX group ID that is used to allow the HPSS servers and utilities to access the HPSS DB2 instance. Use the default value of *hpsssrvr*.

User Name for Servers

The UNIX user name to be used for DB2 authentication. Use the default value of *hpss*.

Create DB2 Instance

By default, this check box is selected to create the DB2 instance.

Instance Owner Password and Verify Password

The UNIX password for the Instance Owner. Use the default prefilled hidden values.

Instance Owner Group

The UNIX group to which the Instance Owner belongs. Default value *hpssdb*.

Owner Home Directory

The directory where the HPSS DB2 instance configuration files are stored. Default directory is `/db2data/db2_hpssdb`.

Instance Client port number

The service port.

Starting Partition Port Number

The initial port number for partition communications. Additional partitions will use additional port numbers following this one. For example, if four partitions are to be configured, and the Starting Partition Port Number is 60000, then 60000-60003 will be used. The available port range should be checked to prevent potential collisions with other applications.

Use SSH or RSH admin

Select SSH or RSH for DB2 partition communications. RSH communications are unsecure, but can provide better performance. The selected mechanism should already be installed and running on the system prior to running configuration in this step. By default, SSH is selected.

Setup DB2 Authentication

By default, this check box is selected to set up DB2 authentication.

Password and Verify Password

The UNIX password used for DB2 authentication. Use the default prefilled hidden values.

Create Config Database

By default, this check box is selected to create the "cfg" database. If this is the initial instance of HPSS and the configuration *CFG* database has not been created, make sure the check box is selected.



If the CFG database has already been created and you do not wish to destroy it, be sure to uncheck this box. Otherwise, it will be destroyed and re-created and any configuration information will be lost.

Config Database Alias

The "real" database is configured as HCFG, while the database alias is the name used by HPSS servers and utilities to reference the database. Do not change the default value.

Primary Log Directory

The location of the cfg database log files. Default location is `/db2data/db2_log/hpssdb/HCFG`. To change the location of the log files, enter the full pathname in the text box.

Mirrored Log

The location of the mirrored log files. It is highly recommended to configure a mirrored log directory. Used to protect information on two separate disks. Default log location is set to `/db2data/db2_logmirror/hpssdb/HCFG`.

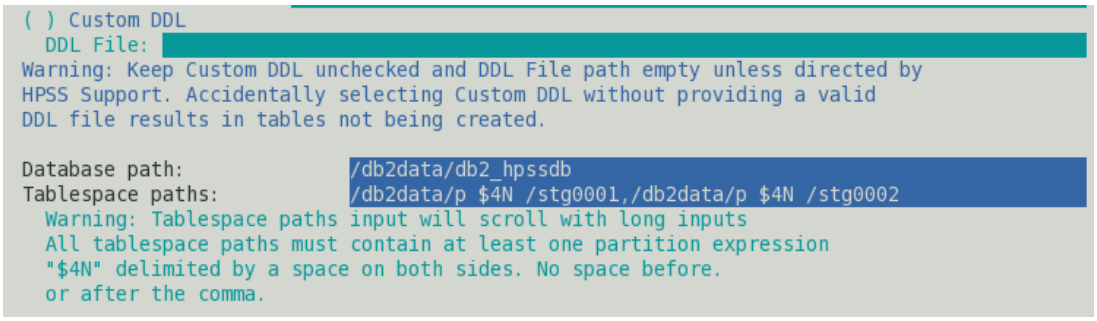
Archived Log 1

The location of the archived log files. It is highly recommended to configure an archived log directory. Used to archive files. Default location is set to `/db2data/db2_logarchive1/`.

Archived Log 2

An alternate location of the archived log files. It is highly recommended to configure an archived log directory. Used to archive files. Default location is set to `/db2data/db2_logarchive2/`.

3. Continuation of DB2 Configuration fields are as displayed on the screen shown:



Custom DDL File

By default, *Custom DDL File* is unchecked with *DDL File* path left empty. This option should only be used under the guidance of HPSS support. This is a special option which can be used to customize the configuration of DB2 tablespaces, buffer pools, and tables. A valid DDL file must exist to utilize this option. IMPORTANT: *Keep default Custom DDL File unchecked and DDL File path empty unless directed by HPSS support. Accidentally selecting Custom DDL File without providing a valid DDL file results in tables not being created.*

Database path

Path to the database instance directory. Default path is `/db2data/db2_hpssdb`.

Tablespace paths

Path to database tablespace directories. No spaces before or after the comma if more than one path is used.
Eg /db2data/stg0001,/db2data/stg0002



All databases in HPSS are configured as automatic storage, whether or not the tablespaces associated with the database are automatic storage or not. Databases that are enabled for automatic storage have a set of one or more storage paths associated with them. If using either automatic storage or DMS

file containers, the tablespaces created to support HPSS will exist under these database paths.

Create Subsys Database

By default the check box to create "subsys" databases is selected.





If the SUBSYS database has already been created and you do not wish to destroy it, be sure to uncheck this box. Otherwise, it will be destroyed and recreated and any existing data will be lost.

Subsys Database Alias Base

The "real" database is configured as HSUBSYS1, while the database alias is the name used by HPSS servers and utilities to reference the database. The default value of *subsys* should normally be used. To support multiple subsystems, this is a *base* value. Each subsystem will have the subsystem number appended to the *base*; for example, if two subsystems were to be created, they would be subsys1 and subsys2. The *$S* symbol can be used to substitute the subsys name (such as subsys1) into a path expression.

Number of Subsystems

The number of subsys databases to configure. This should only be greater than one if the site plans to deploy multiple subsystems. For example, creating two subsystems would result in databases named something like subsys1 and subsys2.

Primary Log Directory

The location of the subsys database log files. To change the location of the log files, enter the full pathname in the text box. The directory `/db2data/db2_log/hpssdb/ $S` is the default setting where $S represents subsys1 based on "Number of Subsystems" equal to 1.

Mirrored Log

The location of the mirrored log files. It is highly recommended to configure a mirrored log directory. Used to protect information on two separate disks. The directory `/db2data/db2_logmirror/hpssdb/ $S` is the default setting.

Archived Log 1

The location of the first archived log files. It is highly recommended to configure an archived log directory. Used to archive files. The directory `/db2data/db2_logarchive1/ $S` is the default setting.

Archived Log 2

The location of the second archived log files. It is highly recommended to configure an archived log directory. The directory `/db2data/db2_logarchive2/ $S` is the default setting.

Extent Size

Used to configure the tablespace extent size.

Tablespace Type

There are three possible options in which to configure the DB2 tablespaces: Automatic Storage, Database Managed (DMS) file containers, and custom Database Definition Language (DDL) scripts.

Automatic Storage

Tablespaces will be created in the storage paths specified during database creation. DB2 manages the container and space allocation for the tablespaces as they are created and populated.

DMS file containers

This configuration works similarly to automatic storage, except that the administrator has the ability to extend the file containers manually. The tablespaces will be created in the storage paths specified during database creation.

Custom DDL File

By default, *Custom DDL File* is unchecked with the *DDL File* path left empty. This option should only be used under the guidance of HPSS support. This is a special option which can be used to customize the configuration of DB2 tablespaces, buffer pools, and tables. A valid DDL file must exist to utilize this option.



Keep default Custom DDL File unchecked and DDL File path empty unless directed by HPSS support. Accidentally selecting Custom DDL File without providing a valid DDL file results in HPSS tables not being created.

Database path

Path to the database instance directory.

Tablespace paths

Path to database tablespace directories.



All databases in HPSS are configured as automatic storage, whether or not the tablespaces associated with the database are automatic storage or not. Databases that are enabled for automatic storage have a set of one or more storage paths associated with them. If using either automatic storage or DMS file containers, the tablespaces created to support HPSS will exist under these database paths.



If using multiple partitions in DB2, make sure to review rsh/ssh configurations that may restrict the connection of the DB2 Instance Owner. For example, if the system has an AllowedUsers section in the file `/etc/ssh/sshd_config`, the DB2 Instance Owner must be listed. When creating partitions directories, make sure it is owned by the DB2 Instance Owner and make sure nothing is inside of it that may interfere with DB2. The $S is used only by **mkhpss** and it means "substitute the subsystem name". It is the only allowable variable for the log file name and is only allowed for the subsystem log file name.

Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_db2_config
```

This will be followed by a timestamp and the duration of the command.

8.6.2.4. *Setting up off-node DB2*

Beginning with version 7.5.2, HPSS supports off-node DB2. This allows sites to scale the DB2 workload independently of the HPSS Core Server system. The **mkhpss** tool does not currently support setting up DB2 in this manner. This section details how the DB2 off-node configuration can be set up.

1. Install DB2 on *all* hosts that will run DB2 (call this the DB2 cluster).

   Note that database partition 0 should always be set up to run on the same host as the HPSS Core Server.

   ```
   # As the ROOT user
   > cd /path/to/DB2/FP8/universal
   > ./db2_install
   <snip>
   ************************************************************
   Install into default directory (/opt/ibm/db2/<DB2 Version>) ? [yes/no]
   yes


   Specify one of the following keywords to install DB2 products.

     SERVER
     CONSV
     EXP
   ```

```
  CLIENT
  RTCL

Enter "help" to redisplay product names.

Enter "quit" to exit.

************************************************************
SERVER
************************************************************
Do you want to install the DB2 pureScale Feature? [yes/no]
no
<snip>
```

Where <DB2 Version> is in the following format: V<version>.<release>. For example:

- V10.5
- V11.1

2. Set up HPSS DB2 Symlink.

```
# As the ROOT user
> cd /opt/ibm/db2
> ln -s <DB2 Version> default
```

3. Install the necessary DB2 license files.

```
# For each license file, as the ROOT user
> /opt/ibm/db2/<DB2 Version>/adm/db2licm -a /path/to/<DB2 Version specific license
file>
> /opt/ibm/db2/<DB2 Version>/adm/db2licm -l
```

4. Make sure that you have an NFS-mounted home directory for the UNIX user that will be your DB2 instance owner. Normally this is the hpssdb user with home directory `/db2data/db2_hpssdb`.

   This means that `/db2data/db2_hpssdb` must be shared across all the nodes that will be part of the DB2 cluster. Also, ensure that each machine in the cluster has root authority on the exported file system by using the "root" option. The rest of the instructions will assume that `/db2data/db2_hpssdb` is the instance owner's home directory and is mounted across all nodes in the DB2 cluster. You also need to ensure that all the relevant DB2 users db2fenc1/hpssdb have the same UID/GID across all the machines in the cluster as well.

5. Create the database instance.

```
# As the ROOT user
> cd /opt/ibm/db2/<DB2 Version>/instance
> ./db2icrt -u db2fenc1 hpssdb
```

6. Set up the db2nodes.cfg file.

   In the following example we have four DB2 hosts:

   hpsscore and *db2host[a-c]*

   (Note that the range string *db2host[a-c]* refers to three separate hosts *db2hosta*, *db2hostb*, and *db2hostc*. This document uses range strings extensively to express examples more compactly).

   hpsscore is where the HPSS Core Server will run and hosts DB2 partition 0.

   db2host[a-c] are the remote DB2 hosts each of which has four DB2 partitions for a total of thirteen database partitions.

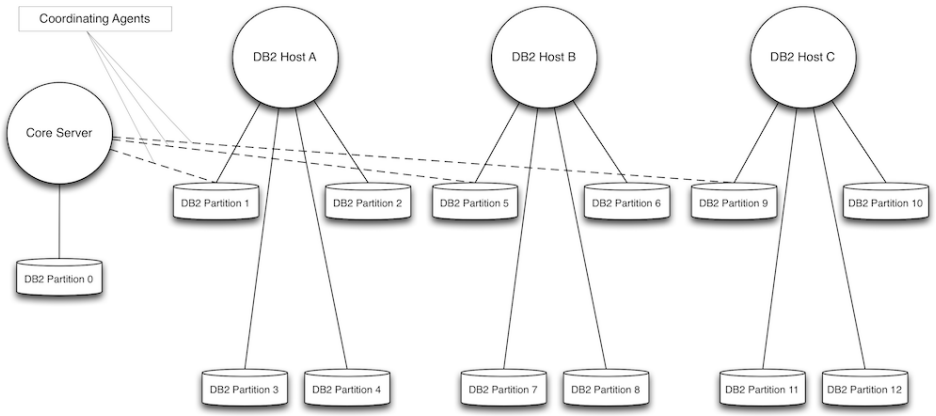


Figure 10. DB2 Off-Node Example

```
> cat /db2data/db2_hpssdb/sqllib/db2nodes.cfg
0  hpsscore 0
1  db2hosta 0
2  db2hosta 1
3  db2hosta 2
4  db2hosta 3
5  db2hostb 0
6  db2hostb 1
7  db2hostb 2
8  db2hostb 3
9  db2hostc 0
10 db2hostc 1
11 db2hostc 2
```

```
12 db2hostc 3
```

7. Set up passwordless ssh between all hosts.
8. Ensure `/etc/services` has the same port numbers allocated to the database instance on each DB2 host. Add service name *db2c_hpssdb* with an assigned port to the `/etc/services` file on each DB2 host as well. This will serve as the port DB2 will listen on for database connection requests. This service name is used to set the database manager SVCENAME parameter (see below....).
9. Create the database filesystems and directory structure.

   We recommend that the filesystems are formatted using the EXT4 filesystem.

   It is also desirable (but not required) to create the database filesystems on a solid-state disk. Ideally, at least the DB2 logs and mirror logs should reside on solid-state.

   Ensure that the instance owner can access all the paths in question.

   Continuing with our example, we set up the following directories on the specified hosts in the example.

   ```
   For DB2 data:
   /db2data/hpssdb/hcfg
   /db2data/hpssdb/hsubsys1
   /db2data/stg[0001-0004]
   /db2data/stg[0001-0004]
   For DB2 primary logs:
   /db2data/db2_log/hpssdb/HCFG
   /db2data/db2_log/hpssdb/HSUBSYS1
   For DB2 mirrorlogs:
   /db2data/db2_logmirror/hpssdb/HCFG
   /db2data/db2_logmirror/hpssdb/HSUBSYS1
   For DB2 primary archive logs:
   /db2data/db2_logarchive1
   For DB2 secondary archive logs:
   /db2data/db2_logarchive1
   For DB2 database backup images:
   /db2data/db2_backup1
   /db2data/db2_backup2
   ```

10. Set the database manager parameters.

    ```
    # As the database instance owner "hpssdb"
    > db2 "update dbm cfg using AUTHENTICATION  SERVER_ENCRYPT"
    > db2 "update dbm cfg using SVCENAME db2c_hpssdb"
    > db2 "update dbm cfg using FEDERATED YES"
    ```

11. Set the DB2 registry variables.

```
# As the database instance owner "hpssdb"
> db2set DB2COMM=TCPIP
> db2set DB2RSHCMD=/usr/bin/ssh
```

12. Start the DB2 instance.

```
# As the database instance owner "hpssdb"
> db2start
```

13. Create the databases.

```
# As the database instance owner "hpssdb"

> db2 "create database hcfg automatic storage yes on \
      '/db2data/stg0001',                    \
      '/db2data/stg0002',                    \
      '/db2data/stg0003',                    \
      '/db2data/stg0004'                     \
      dbpath on '/db2data/db2_hpssdb/HCFG'       \
      pagesize 8192"
DB20000I  The CREATE DATABASE command completed successfully.

> db2 "create database hsubsys1 automatic storage yes on \
      '/db2data/stg0001',                         \
      '/db2data/stg0002',                         \
      '/db2data/stg0003',                         \
      '/db2data/stg0004'                          \
      dbpath on '/db2data/db2_hpssdb/HSUBSYS1'"
DB20000I  The CREATE DATABASE command completed successfully.
```

14. Make sure to set database parameters properly.

    Note the use of range strings below to express the parameters more compactly. They are not intended to be used literally. When executing the commands, make sure to run a command for each element in the range string.

```
# As the database instance owner "hpssdb"

db2 update db config for hcfg DBPARTITIONNUM [0-12] using NEWLOGPATH
/db2data/db2_log/hpssdb/hcfg
db2 update db config for hcfg DBPARTITIONNUM [0-12] using MIRRORLOGPATH
/db2data/db2_logmirror/hpssdb/hcfg
db2 update db config for hcfg DBPARTITIONNUM [0-12] using LOGARCHMETH1
DISK:/db2data/db2_logarchive1
db2 update db config for hcfg DBPARTITIONNUM [0-12] using LOGARCHMETH2
DISK:/db2data/db2_logarchive2
```

```
db2 update db config for hcfg using SELF_TUNING_MEM ON
db2 update db config for hcfg using LOGFILSIZ 25000
db2 update db config for hcfg using LOGPRIMARY 10
db2 update db config for hcfg using LOGSECOND -1
db2 update db config for hcfg using LOCKTIMEOUT 60
db2 update db config for hcfg using SOFTMAX 100
db2 update db config for hcfg using LOGBUFSZ 16384
db2 update db config for hcfg using DATABASE_MEMORY AUTOMATIC
db2 update db config for hcfg using MAXLOCKS AUTOMATIC
db2 update db config for hcfg using LOCKLIST AUTOMATIC
db2 update db config for hcfg using PCKCACHESZ AUTOMATIC
db2 update db config for hcfg using SORTHEAP AUTOMATIC
db2 update db config for hcfg using SHEAPTHRES_SHR AUTOMATIC

db2 update db config for hsubsys1 DBPARTITIONNUM 0      using SELF_TUNING_MEM OFF
db2 update db config for hsubsys1 DBPARTITIONNUM [1-12] using SELF_TUNING_MEM ON
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using NEWLOGPATH
/db2data/db2_log/hpssdb/hsubsys1
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using MIRRORLOGPATH
/db2data/db2_logmirror/hpssdb/hsubsys1
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using LOGARCHMETH1
DISK:/db2data/db2_logarchive1
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using LOGARCHMETH2
DISK:/db2data/db2_logarchive2

db2 update db config for hsubsys1 using LOGFILSIZ 25000
db2 update db config for hsubsys1 using LOGPRIMARY 10
db2 update db config for hsubsys1 using LOGSECOND 20
db2 update db config for hsubsys1 using LOCKTIMEOUT 60
db2 update db config for hsubsys1 using SOFTMAX 100
db2 update db config for hsubsys1 using LOGBUFSZ 16384
db2 update db config for hsubsys1 using DATABASE_MEMORY AUTOMATIC
db2 update db config for hsubsys1 using MAXLOCKS AUTOMATIC
db2 update db config for hsubsys1 using LOCKLIST AUTOMATIC
db2 update db config for hsubsys1 using PCKCACHESZ AUTOMATIC
db2 update db config for hsubsys1 using SORTHEAP AUTOMATIC
db2 update db config for hsubsys1 using SHEAPTHRES_SHR AUTOMATIC
```

15. Back up the databases. When database logging is altered from "circular" to "archival" (setting LOGARCHMETH1/LOGARCHMETH2), it is placed into "BACKUP PENDING" mode and must be backed up before activation is allowed.

```
# As the database instance owner "hpssdb"
> db2 "backup db hcfg on all dbpartitionnums to '/db2data/p \$4N /db2_backup1'"
> db2 "backup db hsubsys1 on all dbpartitionnums to '/db2data/p \$4N /db2_backup1'"
```

16. Activate the databases.

```
# As the database instance owner "hpssdb"
> db2 activate database hcfg
> db2 activate database hsubsys1
```

17. At this point the tablespaces, tables, and database permissions can all be set as before, when all database partitions were running on the same host as the Core Server.

Remote DB2 client access and fileset creation and deletion

This information is pertinent to sites that have chosen to deny remote client access to DB2. The method for configuring DB2 in such a manner is outside the scope of this document; refer to DB2 documentation and support for more information on such a configuration.

If, as part of configuring DB2 to deny remote access, a variable in the DB2 environment called DB2COMM has been unset, the creation and deletion of filesets will fail inside of DB2. You must have a variable named DB2_USE_LOCAL_RESYNC set to the value of *true* when starting DB2 in order for the aforementioned fileset operations to complete successfully:

In csh and tcsh:

```
setenv DB2_USE_LOCAL_RESYNC true
```

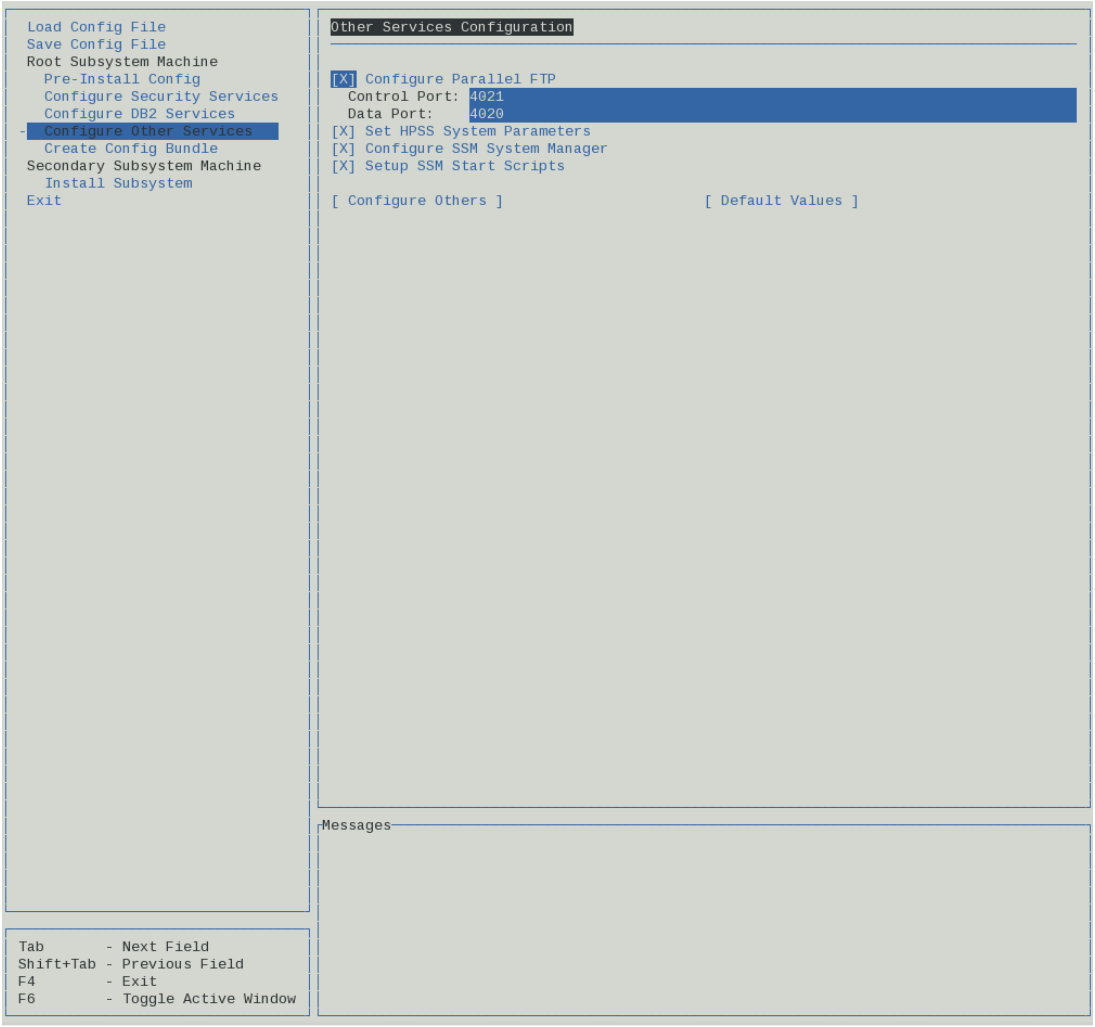
In sh and bash:

```
export DB2_USE_LOCAL_RESYNC=true
```

8.6.2.5. Configure other services

This menu configures various services such as Parallel FTP, HPSS System Parameters, SSM System Manager, and SSM Start Scripts. To configure other services, perform the following steps:

1. From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Other Services" option. The following window will be shown:



2. By default, all boxes are selected. If some items have already been configured, deselect the appropriate box to bypass the reconfiguration of that task. When satisfied with your selections, select the "Configure Others" button and verify the command completes. Depending upon the items selected, HPSS will 1) set up `/etc/services` and inetd to allow HPSS ftp/pftp daemon to be invoked on this system, 2) copy the SSM configuration template files to `/var/hpss/ssm`.
3. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_misc_config
```

This will be followed by a timestamp and the duration of the command.



Failure to *configure other services* will result in SSM failing to start, unless it has already been configured.

1. After exiting **mkhpss**, verify permissions on the generated files. In particular, note the permissions on the keytab files. The `hpss.keytab` is used by HPSS servers to establish credentials. The `mm.keytab` is used by HPSS utility programs. The `kadm5.keytab` is used to establish credentials as the Kerberos admin. Be certain that the permissions on these files allow access only to the appropriate users.

```
/var/hpss/etc/hpss.keytab
/var/hpss/etc/mm.keytab
/var/hpss/krb5kdc/kadm5.keytab
```

8.6.2.6. Configure Logging

HPSS logs to syslog, and by default SSM expects to find logs in `/var/hpss/log/HPSS.log`. You must configure logging before starting SSM for the first time.



If your site uses syslog-ng instead of rsyslog, do not use `configure_logging`. Manually configure syslog-ng using the templates in `$HPSS_ROOT/config/templates/system_files/`.

1. Run the `configure_logging` tool to set up rsyslog and logrotate configurations:

   ```
   /opt/hpss/config/configure_logging
   ```

   This tool will:

   - Install rsyslog configuration to direct HPSS logs to `/var/hpss/log/HPSS.log`
   - Set up log rotation policies
   - Configure the Core Server as a syslog server for remote HPSS nodes

2. Verify the rsyslog configuration was installed:

   ```
   # ls -l /etc/rsyslog.d/hpss.conf
   ```

3. Restart the rsyslog service:

   ```
   # systemctl restart rsyslog
   ```

4. Verify the log directory exists and has proper permissions:

   ```
   # ls -ld /var/hpss/log
   ```

For detailed information about logging configuration options, see Logging overview.



For remote Mover nodes, you will run `configure_logging` with different options as described in [configure-mover-node].

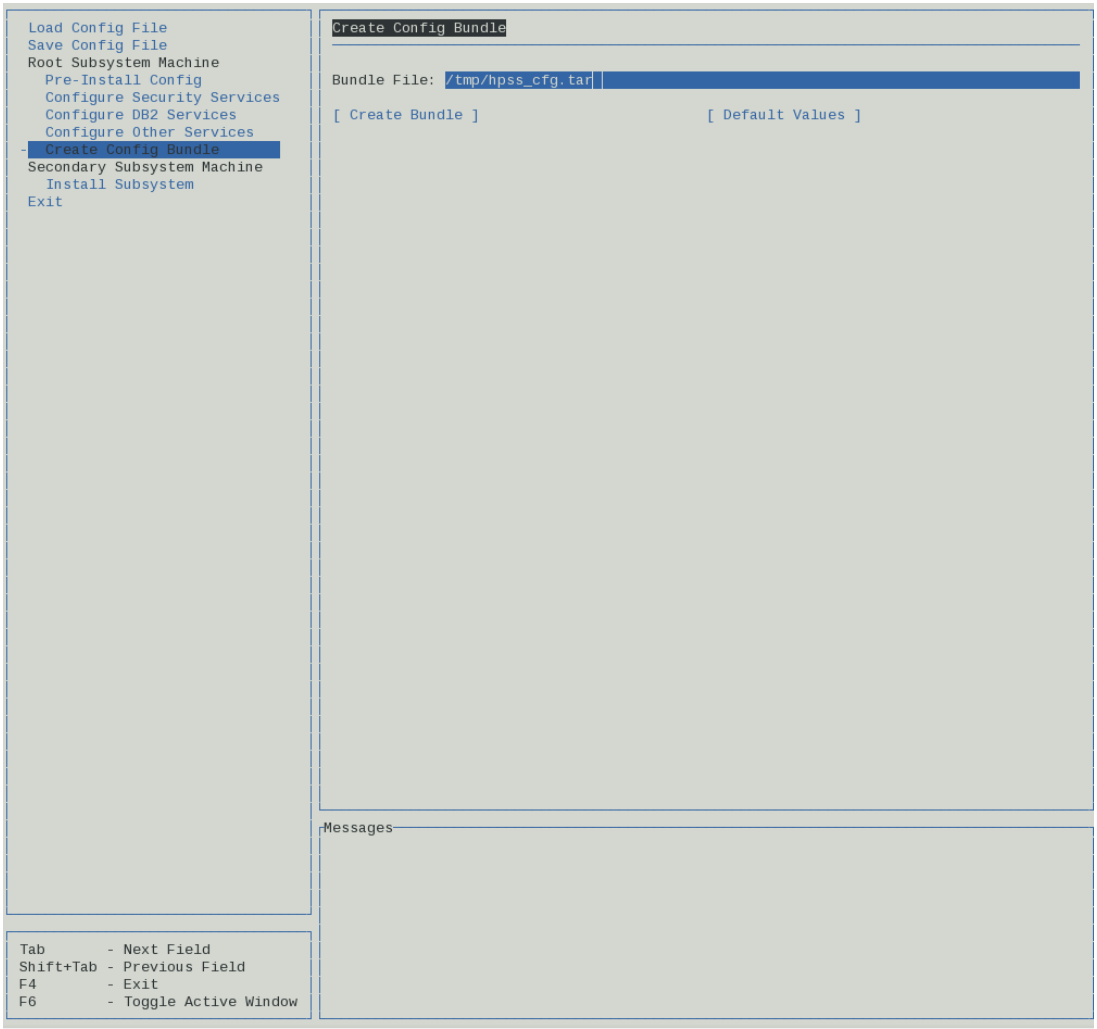
8.6.2.7. Create configuration bundle

To prepare the HPSS configuration information for distribution to other subsystem or Mover nodes,

use the "Create Config Bundle" option on the **mkhpss** (may need to start **mkhpss** if exited previously). This option should be used *only* after the Core Server has been defined and configured by the administrator to include the EndPoint information in the `ep.conf` file. Otherwise, the configuration bundle will not contain the endpoint information and the file will need to be copied or transferred manually to the other nodes after the Core Server is configured.

To create the HPSS configuration bundle perform the following steps:

1. From the "Root Subsystem Machine" submenu in the menu panel, select the "Create Config Bundle" option. The following window will be shown:



2. Modify the "Bundle File" name, if desired.
3. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

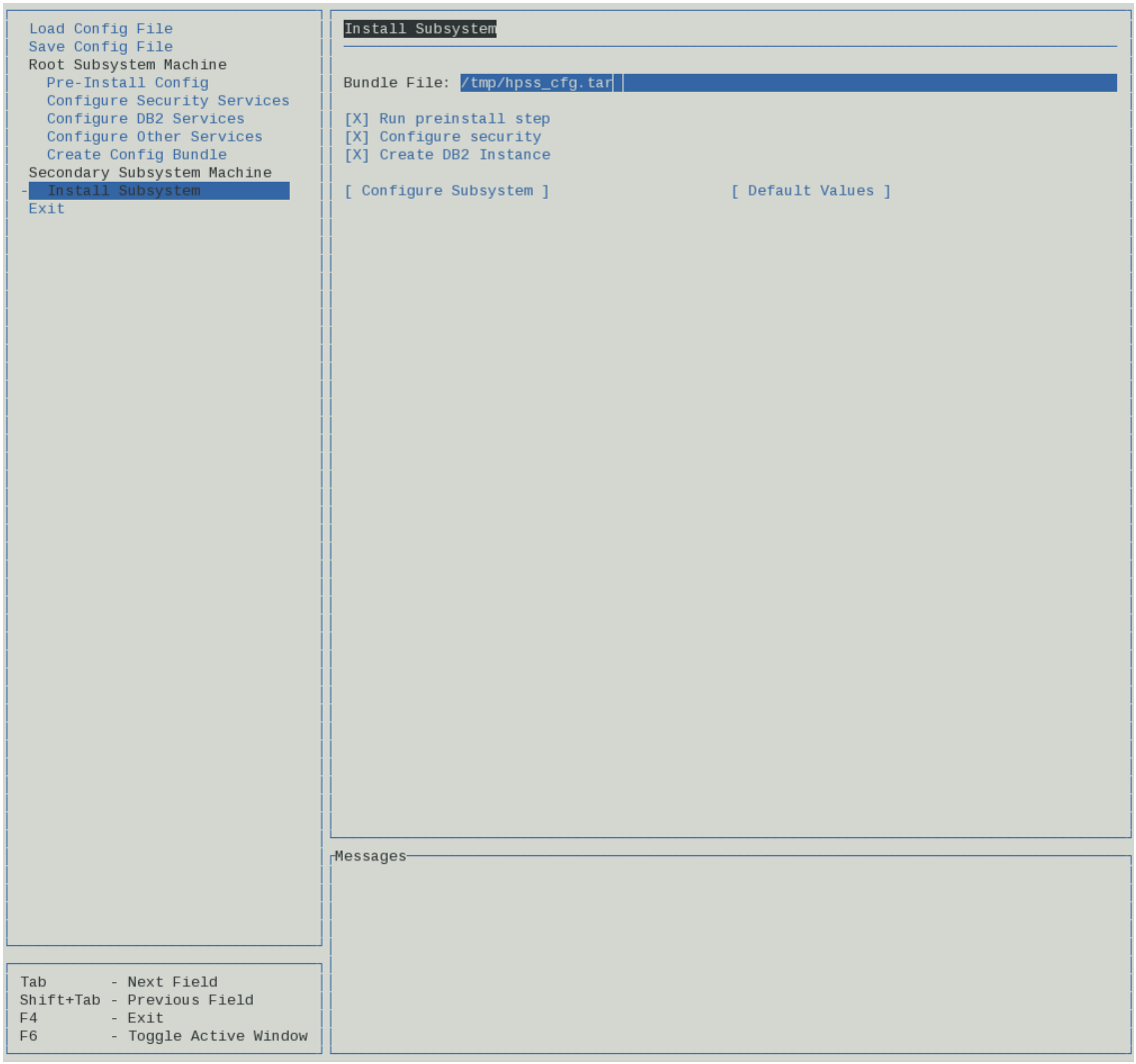
```
run command finished: mkhpss_run_core_create_bundle
```

This will be followed by a timestamp and the duration of the command.

8.6.3. Configure HPSS - secondary subsystem machine

A secondary subsystem machine is a node where a second Core Server is running. For the

secondary subsystem machine, the following configuration steps must be performed. Installing the secondary subsystem requires the same procedure as installing the primary subsystem. So the user can just press on the Configure Subsystem button shown below:



8.6.4. Troubleshooting mkhpss

ERROR1: Before replacing an existing instance with a new one

1. Get rid of the previous User Home Directory (usually `/db2_hpssdb` or `/db2_data/db2_hpssdb`).
2. Delete the db2 instance owner from `/etc/passwd`. (You can check the DB2 instance owner by checking the name of the instance using db2ilist in `/opt/ibm/db2/default`.)
3. Get rid of the existing process using **ps -ef** and grepping for the DB2 instance owner.
4. Get rid of any and all contents such as NODES, mount points, and log files in db2_log, db2_logmirror, db2_logarchives, and tablespaces.
5. Use `/opt/ibm/db2/default/instance/db2idrop` to drop the previous instance.

ERROR2 : Updating an existing instance

1. Get rid of the database created in the database paths before running **mkhpss**.
2. There is no need to tamper with the logs. But if the old logs are needed make sure to specify new paths for logging, mirroring, and archiving.

ERROR3 : Services/port error

1. Clear the services that contain the DB2 Instance Owner in `/etc/services`.

ERROR4 : Enable Auto-Configuration Default

1. Get rid of the User Home Directory before running the DB2 portion of **mkhpss**.
2. Make sure **mkhpss** created the User Home Directory under the ownership of the DB2 Instance Owner.

8.7. Post-installation procedures

After the HPSS software has been installed and its infrastructure has been configured, perform the following verifications:

1. Assuming default installation and configuration, verify the following directories have been created:

   `/opt/hpss/bin/`<HPSS binaries>

   `/opt/hpss/lib/`<HPSS libraries>

   `/opt/hpss/include/`<HPSS include and idl files>

   `/opt/hpss/tools/`<HPSS tools>

   `/opt/hpss/man/`<HPSS man pages>

   `/opt/hpss/config/`<HPSS configuration scripts>

   `/opt/hpss/`<HPSS source files> *Included only if the hpss-src package is installed.*

   `/var/hpss/`<HPSS configuration files>

2. Verify that the HPSS file ownerships and file permissions are set as follows:

Executable files	`rwxr-xr-x root root`
Include files	`rw-r--r-- root root` (write permission on generated files)
Library files	`rwxr-xr-x root root`
Source files	`rw-r--r-- root root`
Make files	`rw-r--r-- root root`



In particular, note that permissions on `/var/hpss/etc/mm.keytab` control the access to HPSS from many utility programs. Any user who can read `mm.keytab` will have permission to read and write directly into the HPSS database.

3. Verify that the DB2 permanent license has been set up by issuing the following commands:

```
% su -
% /opt/IBM/db2/default/adm/db2licm -l
```

Refer to Install DB2 and set up permanent license for more information on how to set up the DB2 license for an HPSS system.

8.8. Locate HPSS documentation and set up manual pages

This section describes the HPSS documentation provided to aid in the administration of the HPSS system as well as providing the basis for SSM help facility. It also describes the procedure to set up manual pages for HPSS utilities.

8.8.1. Documentation and SSM help package

The HPSS documentation is available via the HPSS website as a tar file and in PDF format. The HPSS documentation includes the following:

- *HPSS Admin Guide*
- *HPSS Error Manual*
- *HPSS Programmer's Reference*
- *Programmer's Reference - I/O Supplement*
- *HPSS User's Guide*

The HTML version of the *HPSS Admin Guide* also serves as the source for the SSM Help feature. These HTML files must be accessible from each host from which the SSM hpssgui(7) program is executed in order to be available in the SSM Help feature. The hpssuser(7) program can be used to bundle the HTML files for delivery to the hpssgui(7) host machine. The recommended installation location for the HTML files on each hpssgui(7) host is `/opt/hpss/ssmhelp` for Linux platforms and `c:\hpss\doc` for Windows platforms.

8.8.2. Manual pages setup

Perform one of the following steps to set up HPSS manual pages:

- Edit the `/etc/man_db.conf` file. Assuming that the HPSS tree is installed under `/opt/hpss`, add the following line in the section with other MANDATORY_MANPATH lines:

```
MANDATORY_MANPATH /opt/hpss/man
```

- Edit a system file like `/etc/profile.d/hpss.sh` to ensure that the MANPATH environment variable includes the HPSS man pages directory. This can be done with the following:

```
% su - root
```

```
% vi /etc/profile.d/hpss.sh
Add the following and save the file:
if ! manpath | /bin/grep -q /opt/hpss/man; then
export MANPATH="/opt/hpss/man:"
fi
```

- Edit a system file like `/etc/profile.d/hpss.sh` to ensure that the PATH environment variable is set to include the `/opt/hpss/bin` directory. This can be done with the following:

```
% su - root
% vi /etc/profile.d/hpss.sh
Add the following and save the file:
if ! echo ${PATH} | /bin/grep -q /opt/hpss/bin; then
PATH=/opt/hpss/bin:${PATH}
fi
```

After one of the above is done, users who subsequently log in to the system should be able to view HPSS manual pages. The command **manpath** should show a directory containing `/opt/hpss/man`, and users should be able to view manual pages on HPSS commands or files. For example, to view information on lshpss, run the following:

```
% man lshpss
```

This document also contains the man pages for ease of reference. For example, the link lshpss(7) refers to the lshpss man page.

8.9. Define HPSS environment variables

While most, if not all HPSS environment variables can be used as defined by HPSS, they should be reviewed to ensure that they are set to reflect your environment. The HPSS environment variables and their default values are defined in `/opt/hpss/include/hpss_env_defs.h` file. See Appendix E: HPSS Environment Variables.

which lists the default values of all HPSS environment variables. These environment variables may be overridden in `/var/hpss/etc/env.conf` or in the system `/etc/environment` file.

The **/opt/hpss/config/hpss_set_env** utility can be used to insert environment variables into the `/var/hpss/etc/env.conf` file, list the values of the current HPSS environment variables, or list all HPSS default environment values:

Usage:

```
hpss_set_env [-all] [-def] [-set ENVNAME=NEWVALUE] |
                            ENVNAME [ENVNAME...]]
```

Where:

```
-all         Show current HPSS environment values
-def         Show all HPSS default environment values
-set         Set HPSS default environment values
ENVNAME      Display current value for ENVNAME
```

8.10. Set up a remote PVR

To set up a Remote PVR, the following steps must be completed:

1. **Initially configure the remote system**

   The remote system running the PVR should have the same authentication files available as the Core Server system. For example, if UNIX is being used with HPSS password files, the files in `/var/hpss/etc` should be copied over to the remote system. If UNIX password files are being used, care should be taken that the same passwords, users, and groups exist on the remote system.

   Further, the remote system should have all HPSS prerequisite software installed as described in the Release Notes.

2. **Install hpss-lib and hpss-pvr RPMs on the remote system**

   ```
   # rpm -ivh hpss-lib-X.Y-0.el8.x86_64.rpm hpss-pvr-X.Y-0.el8.x86_64.rpm
   ```

3. **Install and configure a Db2 client on the remote system**

   a. Create an "hpssdb" user

      Use the same hpssdb GID and UID from the core server.

      ```
      # mkdir /var/hpss
      # groupadd --gid <hpssdb gid> hpssdb
      # useradd --create-home --home <hpssdb homedir> --uid <hpssdb uid> \
        --gid <hpssdb gid> --shell /bin/bash hpssdb
      ```

   b. Create a Db2 instance

      ```
      # /opt/ibm/db2/vx.x/instance/db2icrt -a CLIENT -s client -u hpssdb hpssdb

      Where -a is authentication, -s is instance type, -u is for ID and
      instance name.
      ```

   c. Set DB2COMM

      Must be done as hpssdb.

```
# su - hpssdb
# db2set DB2COMM=tcpip
```

d. Verify the local service in `/etc/services` for DB2 support

Copy the DB2 service entries from the Core Server `/etc/services`.

```
# Local services
db2c_hpssdb    59999/tcp
DB2_hpssdb     60000/tcp
DB2_hpssdb_1   60001/tcp
DB2_hpssdb_2   60002/tcp
DB2_hpssdb_END 60003/tcp
```

e. Catalog the tcpip node

```
# db2 catalog tcpip node CORSVR remote <core server hostname> server db2c_hpssdb
```

f. Catalog the database hcfg

```
db2 catalog db hcfg as cfg at node CORSVR
```

g. Verify that Db2 client can connect to the Db2 server on the HPSS core machine, list out the table for cfg/hcfg and verify the contents of a few known Db2 tables. The results should be identical to the HPSS Core Server.

```
  # db2 connect to cfg user hpssdb using hpss
  # db2 list tables for schema hpss
  # db2 "select * from hpss.COS"
```

4. **Configure logging**

Configure logging using the `/opt/hpss/config/configure_logging` tool.

```
/opt/hpss/config/configure_logging -s remote -H <core server hostname>
```

5. **Configure security**

Configure security using the `/opt/hpss/config/setup_pam.pm` tool. Alternatively, /etc/pam.d/hpss can be copied from the Core Server.

```
/opt/hpss/config/setup_pam.pm
[ setting up PAM config ]
```

```
[ using local HPSS passwords ]
```

HPSS relies on the standard Linux GSS for security. Copy the `/opt/hpss/config/templates/system_files/mech.hpss.conf` file to `/etc/gss/mech.d/`. This is part of the RPM install process.

1. **Configure an HPSS Startup Daemon for the remote system**

   In SSM, configure a new HPSS Startup Daemon. The Execute Hostname field should be the remote system.

2. **Configure a SCSI PVR for the remote system**

   In SSM, configure a new SCSI PVR. The Execute Hostname field should be the remote system.

When you start HPSS, you will need to start the startup daemon by running the following command on the remote system:

```
/opt/hpss/bin/rc.hpss -d
```

You can then monitor and administer the remote SCSI PVR.

Once you have set up a single remote SCSI PVR, adding additional remote SCSI PVRs on the same remote system can be done via SSM with no additional setup required.

8.11. Tune DB2

Database tuning is an important aspect of maximizing HPSS performance, but it is a complex topic, requiring study and experimentation, to do well. **mkhpss** creates initial configuration parameters for HPSS DB2 instances that we find to be effective for most systems, but additional tuning to deal with the specifics of a given HPSS installation may improve database performance. Administrators wishing to learn more about DB2 tuning are referred to the *HPSS DB2 Tuning Guide*, available from HPSS support, the *DB2 Administration Guide: Performance*, available online from the IBM DB2 website, the IBM *"Database Fundamentals >> Performance tuning"* section of the DB2 InfoCenter. and the many after-market books available on this subject.

Be sure to contact HPSS support for advice before making changes to DB2 configuration parameters. See also the Backup and recovery chapter for additional information.

8.12. Supporting both UNIX and Kerberos authentication for SSM



This section applies to authenticating with Java SSM. For SSM API services, OIDC is used to authenticate users with the SSM API and can be federated to use Kerberos.

Once security services have been configured for your system (see Section Configure HPSS security services for details), if both UNIX and Kerberos have been set up, it is possible to configure your

system to support both UNIX and Kerberos authentication for the SSM. If you can log in to the system using the SSM GUI, you can use it to set this up, as described in the [configuring-system-manager-authentication-for-ssm-clients] section. If that is not an option because no available authentication mechanism is configured, you can use the procedure that follows to set up support for both authentication mechanisms. The combination of UNIX authentication with LDAP authorization is not supported at this time, so it only makes sense to do this if you are using UNIX authorization.

To set up support for both authentication mechanisms, change the following fields in DB2:

Table : Field	Old	New	Where
server : NUM_AUTH_MECHS	1	2	desc_name = 'SSM System Manager'
server : AUTHN_MECHS1_MECHANISM	0	2	
server : AUTHN_MECHS1_AUTH_TYPE_KEY	0	1	
serverinterfaces : AUTHN_MECH_SET_NUM_MECHS	1	2	server_id = (select server_id from server where desc_name = 'SSM System Manager\') and descriptive_name = 'Administrative Client Interface'
serverinterfaces : AUTHN_MECH_SET_MECHS1	0	2	

This can be accomplished using the **db2** interactive utility. Here's a sample session showing the commands to use. You'll need to be logged in to UNIX as user *hpss* to have the needed permissions on the database.

```
$ db2
(c) Copyright IBM Corporation 1993,2002
Command Line Processor for DB2 SDK 8.2.5
...
For more detailed help, refer to the Online Reference Manual.

db2 => connect to hcfg

   Database Connection Information

 Database server = DB2/LINUX 8.2.5
 SQL authorization ID = HPSS
 Local database alias = HCFG

db2 => set schema hpss

db2 => select num_auth_mechs, authn_mechs1_mechanism,
authn_mechs1_auth_type_key
 from server where desc_name = 'SSM System Manager'
```

```
NUM_AUTH_MECHS AUTHN_MECHS1_MECHANISM AUTHN_MECHS1_AUTH_TYPE_KEY
-------------- ---------------------- --------------------------
             1                      0                          0

  1 record(s) selected.

db2 => select authn_mech_set_num_mechs, authn_mech_set_mechs1 from
serverinterfaces where server_id = (select server_id from server where
desc_name = 'SSM System Manager') and descriptive_name = 'Administrative
Client Interface'

AUTHN_MECH_SET_NUM_MECHS AUTHN_MECH_SET_MECHS1
------------------------ --------------------
                       1                    0

  1 record(s) selected.

db2 => update server set (num_auth_mechs, authn_mechs1_mechanism,
authn_mechs1_auth_type_key) = (2, 2, 1) where desc_name = 'SSM System
Manager'
DB20000I The SQL command completed successfully.

db2 => update serverinterfaces set (authn_mech_set_num_mechs,
authn_mech_set_mechs1) = (2, 2) where server_id = (select server_id from
server where desc_name = 'SSM System Manager') and descriptive_name =
'Administrative Client Interface'
DB20000I The SQL command completed successfully.

db2 => select num_auth_mechs, authn_mechs1_mechanism,
authn_mechs1_auth_type_key
 from server where desc_name = 'SSM System Manager'

NUM_AUTH_MECHS AUTHN_MECHS1_MECHANISM AUTHN_MECHS1_AUTH_TYPE_KEY
-------------- ---------------------- --------------------------
             2                      2                          1

  1 record(s) selected.

db2 => select authn_mech_set_num_mechs, authn_mech_set_mechs1 from
serverinterfaces where server_id = (select server_id from server where
desc_name = 'SSM System Manager') and descriptive_name = 'Administrative
Client Interface'

AUTHN_MECH_SET_NUM_MECHS AUTHN_MECH_SET_MECHS1
------------------------ --------------------
                       2                    2

  1 record(s) selected.

db2 => terminate
```

```
DB20000I The TERMINATE command completed successfully.
$
```

If you're using the local HPSS password file (HPSS_UNIX_USE_SYSTEM_COMMANDS=FALSE), you need to make sure it contains entries for users *root* and *hpss*.

Now you can use either security mechanism ("unix" or "krb5") in your SSM configuration file.

8.13. HPSS IPv6 support

The people that designed IPv4 never imagined that we would need more than 2^{32} IP addresses. Nearly every computing device made today has at least one network interface. When people began to realize that there were not enough IPv4 addresses for our needs, a new protocol, IPv6, was developed that would use a 128-bit addressing scheme. There are 3.4×10^{38} IPv6 addresses. As a comparison, there are roughly 9×10^{21} stars in the observable universe. The main difficulty is to get everyone out there buying new hardware that supports the new protocol. In February 2011, the last of the major IPv4 blocks were given out. This should be incentive enough to get people to begin using IPv6.

HPSS has been updated to allow usage of IPv6 networks. In order to determine what protocol HPSS will use, the system administrator must set the following in the `env.conf` file:

Table 16. Protocol settings

Environment variable	Value	Meaning
HPSS_NET_FAMILY	ipv4_only	Only allow use of IPv4 (this is the default).
HPSS_NET_FAMILY	ipv6	Allow usage of both IPv4 and IPv6. Note that if IPv6 is available, HPSS will prefer it over IPv4.
HPSS_NET_FAMILY	ipv6_only	Only allow use of IPv6.

8.13.1. Usage examples

HPSS_NET_FAMILY=ipv4_only

This is the HPSS default. HPSS services will only allow the use of the IPv4 protocol.

```
> rpcinfo -s
   program version(s) netid(s)                         service     owner
 536870913  1         tcp                              -           superuser
 536870916  1         tcp                              -           superuser
 536870915  1         tcp                              -           superuser
 536870917  1         tcp                              -           superuser
 536870920  1         tcp                              -           superuser
 536870914  1         tcp                              -           superuser
 536870912  1         tcp                              -           superuser
```

```
 536870918  1         tcp                             -           superuser
 536870919  1         tcp                             -           superuser

> lsof -P -n | grep "hpss_core 22743 " | grep LISTEN
hpss_core 22743 root 14u IPv4 296030 0t0 TCP *:42580 (LISTEN)

> nc -v -4 10.0.2.15 42580
Connection to 10.0.2.15 42580 port [tcp/*] succeeded!

> lsof -P -n | grep "nc " | grep IP
nc 24120 root 3u IPv4 366364 0t0 TCP
10.0.2.15:39038->10.0.2.15:42580 (ESTABLISHED)

> lsof -P -n | grep "hpss_core 22743 " | grep 39038
hpss_core 22743 root 15u IPv4 366365 0t0 TCP
10.0.2.15:42580->10.0.2.15:39038 (ESTABLISHED)
```

HPSS_NET_FAMILY=ipv6

HPSS services will prefer to use the IPv6 protocol, if available.

```
> rpcinfo -s
   program version(s) netid(s)                         service     owner
 536870913  1         tcp,tcp6                        -           superuser
 536870916  1         tcp,tcp6                        -           superuser
 536870915  1         tcp,tcp6                        -           superuser
 536870917  1         tcp,tcp6                        -           superuser
 536870920  1         tcp,tcp6                        -           superuser
 536870914  1         tcp,tcp6                        -           superuser
 536870912  1         tcp,tcp6                        -           superuser
 536870918  1         tcp,tcp6                        -           superuser
 536870919  1         tcp,tcp6                        -           superuser

> lsof -P -n | grep "hpss_core 12586" | grep LISTEN
hpss_core 12586 root 14u IPv6 163012 0t0 TCP *:35941 (LISTEN)

> nc -v -4 10.0.2.15 35941
Connection to 10.0.2.15 35941 port [tcp/*] succeeded!

> lsof -P -n | grep "nc " | grep IP
nc 21611 root 3u IPv4 285809 0t0 TCP
10.0.2.15:51191->10.0.2.15:35941 (ESTABLISHED)

> lsof -P -n | grep "hpss_core 12586 " | grep 51191
hpss_core 12586 root 15u IPv6 285810 0t0 TCP
10.0.2.15:35941->10.0.2.15:51191 (ESTABLISHED)

> nc -v -6 ::1 35941
Connection to ::1 35941 port [tcp/*] succeeded!
```

```
> lsof -P -n | grep "nc " | grep IP
nc 27208 root 3u IPv6 466915 0t0 TCP
[::1]:33645->[::1]:35941 (ESTABLISHED)

> lsof -P -n | grep "hpss_core 12586 " | grep 33645
hpss_core 12586 root 15u IPv6 466916 0t0 TCP
[::1]:35941->[::1]:33645 (ESTABLISHED)
```

HPSS_NET_FAMILY=ipv6_only

HPSS services will only allow the use of the IPv6 protocol.

```
> rpcinfo -s
   program version(s) netid(s)                         service     owner
 536870913  1         tcp6                             -           superuser
 536870916  1         tcp6                             -           superuser
 536870915  1         tcp6                             -           superuser
 536870917  1         tcp6                             -           superuser
 536870920  1         tcp6                             -           superuser
 536870914  1         tcp6                             -           superuser
 536870912  1         tcp6                             -           superuser
 536870918  1         tcp6                             -           superuser
 536870919  1         tcp6                             -           superuser

> nc -v -4 10.0.2.15 57947
nc: connect to 10.0.2.15 port 57947 (tcp) failed: Connection refused
```

Chapter 9. Installation and configuration of the Elastic (ELK) Stack

9.1. Installing the ELK stack

This section explains how to install the various parts of the ELK stack. The parts are:

- Filebeat
- Logstash
- Elastic
- Kibana

The HPSS 10.1 ELK dashboards and templates, were developed using ELK version 7.15.0.

9.1.1. Install Filebeat

Download Filebeat from https://www.elastic.co/downloads/beats/filebeat.

On the right side, select "View past releases" and then version 7.15.0.

Use the Filebeat installation instructions found on the download page, with the following modifications:

- Install Filebeat on the HPSS Core Server
- Use the **filebeat-hpss.yml** provided in the HPSS RPM instead of the default **filebeat.yml**.
- Update the **filebeat-hpss.yml** to point to the correct Logstash



The provided Filebeat template assumes Logstash is installed on the same system. However, the best practice is to run Logstash on a separate system. Change the stanza below to contain the correct host for Logstash.

```
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"] # Update this if Logstash will be run on a different host
  enabled: true
```

It is recommended that you start Filebeat as a service using systemd. You can find Filebeat setup instructions at https://www.elastic.co/guide/en/beats/filebeat/current/running-with-systemd.html

If you need a sample services file, use the following:

```
[Unit]
Description=Filebeat
```

```
[Service]
User=root
WorkingDirectory=<directory where Filebeat was installed>
ExecStart=filebeat -c filebeat-hpss.yml
Restart=always

[Install]
WantedBy=multi-user.target
```

9.1.2. Install Logstash

Download Logstash from https://www.elastic.co/downloads/logstash

On the right side, select "View past releases" and then version 7.15.0.

Use the Logstash installation instructions found on the download page, with the following modifications:

- Use the **logstash-hpss.conf** instead of the default **logstash.yml**.

It is recommended that you start Logstash as a service using systemd. While there are no guides for setting up Logstash with systemd, it does not differ from other processes. You can find instructions on running Logstash at https://www.elastic.co/guide/en/logstash/current/setup-logstash.html

9.1.3. Install Elastic

Download Elastic from https://www.elastic.co/downloads/elasticsearch

On the right side, select "View past releases" and then version 7.15.0.



It is recommended that you do **NOT** run Elastic on the HPSS Core Server.

Use the Elastic installation instructions found on the download page with the following modifications:

There are no HPSS-specific Elastic settings, except for the authentication section of the logstash template `logstash-hpss.conf`. This section will need to be modified for proper authentication - but the index should not be changed.

```
elasticsearch {
   hosts => ["http://localhost:9200"]
   index => "%{msgid}-%{[@metadata][version]}-%{+YYYY.MM}"
   user => "elk"
   password => "elk1elk1"
}
```

To start Elastic automatically, see the instructions at https://www.elastic.co/guide/en/elasticsearch/reference/current/starting-elasticsearch.html



By default, security is disabled. Depending on how the logstash / elastic instance is being accessed, update the `logstash-hpss.conf template` with your authentication options. Use filebeat to test that metrics are making it to the elasticsearch instance.

Refer to the elastic product documentation for specific guidance on setting up a secure elastic installation.

9.1.4. Install Kibana

Download Kibana from https://www.elastic.co/downloads/kibana

On the right side, select "View past releases" and then version 7.15.0.



It is recommended that you do **NOT** run Kibana on the HPSS Core Server.

Use the Kibana installation instructions found on the download page, with the following modifications:

- Load the HPSS Dashboards using Load the HPSS templates into Kibana.

To start Kibana automatically, see the instructions at https://www.elastic.co/guide/en/kibana/current/start-stop.html

9.1.5. Scaling Elastic

The configuration provided in the above links are for a very basic installation of Elastic. As your system grows, you will need to scale it to meet the increased demand. The Elastic website has a wealth of information on how to scale your system. Here are two web sites to get you started on scaling Elasticsearch:

- https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html
- https://www.elastic.co/blog/small-medium-or-large-scaling-elasticsearch-and-evolving-the-elastic-stack-to-fit

9.2. Installing the HPSS data capture components

9.2.1. HPSS data capture scripts

The HPSS data capture components are a series of scripts that call HPSS tools and translate the output into a format usable by ELK. They are designed to be run as cron jobs.

- hpssadm_hs_info
- hpssadm_devices_and_drives_list
- hpssadm_sc_list
- hpssadm_servers_list
- hpss_dump_acct

- hpss_dump_server_metrics
- hpss_dump_db_metrics



The HPSS data capture scripts are included with the HPSS Core Server RPM.

9.2.2. How to configure the data capture scripts

The data capture scripts should be configured as cron jobs. The arguments for the data capture scripts will vary based on system authentication.

Identify the HPSS user and credential desired, and verify that the capture scripts run successfully. The user's default keytab will be used by default, but can be specified using the -U (user) and -k (keytab) parameters. Consult the man pages and usage for the particular tools.

Once the configuration is complete, as the HPSS user, run **crontab -e** and add entries like the following:

```
*/5 * * * * /opt/hpss/bin/hpss_dump_server_metrics -r <site specific args>
*/5 * * * * /opt/hpss/bin/hpss_dump_db_metrics -r <site specific args>
*/5 * * * * /opt/hpss/bin/hpss_dump_acct -r <site specific args>
*/5 * * * * /opt/hpss/bin/hpssadm_hs_info -r <site specific args>
*/5 * * * * /opt/hpss/bin/hpssadm_sc_list -r <site specific args>
*/5 * * * * /opt/hpss/bin/hpssadm_servers_list -r <site specific args>
*/5 * * * * /opt/hpss/bin/hpssadm_devices_and_drives_list -r <site specific arg>
```

This is an example - these tools need to be run periodically, but can be configured based on site needs.

This configuration will capture all the statistics every five minutes. It is recommended that sites decide how frequently to capture data and modify the crontab to run the capture scripts based on that need. For example, a site might need to capture the server status more frequently and the account summary data less frequently.



hpss_dump_db_metrics queries a specific subsystem database. Without the -s argument specified the HSUBSYS1 is queried by default. To query multiple subsystems the hpss_dump_server_metrics script would need to be called multiple times in the crontab.



If you change the frequency of data capture, you will need to change some of the provided Kibana templates. These lenses can be identified by the "Last 5 minutes" tag in the upper right hand corner when looking at the dashboard. For those lenses, the "Customized Time Range" for the lens needs to match the frequency of data collection. Otherwise, the lens will summarize too much information.

9.3. Setup the HPSS dashboards

9.3.1. Load the HPSS templates into Kibana

To load the HPSS templates into Kibana:

1. Create a space named HPSS in Kibana
2. Import the hpss/tools/capture_scripts/templates/hpss_kibana_templates.ndjson file

The instructions for creating a space and importing objects can be found at the following:

- https://www.elastic.co/guide/en/kibana/master/xpack-spaces.html
- https://www.elastic.co/guide/en/kibana/current/managing-saved-objects.html



The HPSS dashboard templates are included with the HPSS Core Server RPM.

9.3.2. The HPSS dashboards

The following dashboards will be imported:

Dashboard Name	Description
HPSS SSM Dashboard	Displays information from the HPSS GUI window.
Heatmaps: Transfers by COS	Heatmaps that show the activity by HPSS COS.
Heatmaps: Transfers by SC	Heatmaps that show the activity by HPSS SC.
Heatmaps: Transfers by users	Heatmaps that show the activity by HPSS user.
Status: Core Server specific information	Detailed statistics about the core server.
Status: Devices and drives	Current and status history for devices and drives information from the HPSS SSM.
Status: HPSS Servers	Current and status history for the HPSS servers from the HPSS SSM.
Status: HPSS Servers - Op State	Current and history for the HPSS server's Op State from the HPSS SSM.
Status: Storage Classes	Current and status history for the active Storage Classes from the HPSS SSM.
Status: Tape Devices	Shows information about the PVL jobs and tape mounts.
Storage: Broken down by COS	Displays the latest values for file counts and bytes stored broken down by COS.
Storage: Broken down by SC	Displays the latest values for file counts and bytes stored broken down by SC.

Storage: Summary	See summaries of disk and tape storage spaces. Also provides tape mounts per hour.
Transfers: Broken down by COS	Transfer information on reads and writes broken down by COS. This includes both the bytes transferred and the number of transfers.
Transfers: Broken down by device	Bytes written, read, and errors by device.
Transfers: Broken down by mover	Transfer information on reads and writes broken down by HPSS mover. This includes both the bytes transferred and the number of transfers.
Transfers: Operational statistics	Shows the rate of creates, deletes, and opens occurring on the system.

9.4. Captured data

The captured data is stored as indexes in ELK. The index names are based on the message id (msgid field), the year and month, and the ELK version. Here are the msgids produced by HPSS:

hpss_hs

This index contains the output from the **hpssadm "health_status info"** command. Some of this data will reset when the SSM restarts.

hpss_sc_active

This index contains the output from the **hpssadm "list Active Storage Classes"** command. The space values here can not be added together because Disk Storage Classes (SC) use bytes and Tape SCs use virtual volumes (VV). It is possible to write custom filters to group just disk or tape SC.

hpss_devices_and_drives

This index contains the output from the **hpssadm "list Devices and Drives"** command. The "Bytes Read" and "Bytes Written" fields are stripped from this output. This data resets when the SSM resets. It is not reset by **hpssadm_dump_devices_and_drives**.

hpss_servers

This index contains the output from the **hpssadm "list Servers"** command.

hpss_acct_sum

This index contains the output from the **dump_acct_sum -v** command. The data here increments and does not reset when the servers go down.

hpss_acct_bandwith

This index contains the output from the **dump_acct_sum -b** command. The data here increments and does not reset when the servers go down.

hpss_cos_file_count

This index contains the output from the dump_acct_sum(7) command. The data here increments

and does not reset when the servers go down. This data isn't used on any dashboard. The provided templates calculated the similar values by summing the **hpss_acct_sum -v** command output.

hpss_server_metrics_mover

This index contains the output from **hpss_server_metrics -t mover** command. The data fields reset after every call to **hpss_dump_server_metrics** are:

- RequestProcessed
- DataTransfers
- RequestErrors
- BufferSize
- BytesMoved
- TotalMoveTime

hpss_server_metrics_device_stats

This index contains the output from the **hpss_server_metrics -t drives** command. The data fields reset after every call to **hpss_dump_server_metrics** are:

- NumberOfErrors
- BytesRead
- BytesWritten

hpss_server_metrics_storage

This index contains the output from the **hpss_server_metrics -t storage** command. No fields are reset in this index.

hpss_server_metrics_config

This index contains the output from the **hpss_server_metrics -t config** command. No fields are reset in this index.

hpss_server_metrics_statfs

This index contains the output from the **hpss_server_metrics -t statfs** command. No fields are reset in this index.

hpss_server_metrics_core

This index contains the output from the **hpss_server_metrics -t core** command. All the data fields reset after every call to **hpss_dump_server_metrics**.



There are too many fields (approximately 100) to list every one in this document. All of them are reset by the calls to **hpss_dump_server_metrics**.

hpss_server_metrics_core_limits

This index contains the output from the **hpss_server_metrics -t limits** command. No fields are reset in this index.

Chapter 10. HPSS Rumbler

10.1. Introduction

The Rumbler feature is an HPSS configuration option of Db2 along with a set of tools to capture and report changes to the HPSS namespace during a given timeframe. The resulting outputs can be used to feed other applications that monitor usage of the archive and aid in managing large unstructured data repositories.



Commands with a preceding "$" prompt are to be executed by the Db2 instance owner *hpssdb*. Commands with a preceding "#" are to be executed under *root*.



The *ibm_db* and *pycrypto* Python must be installed and usable by the *hpssdb* account. *A Python development environment is required to build and install the ibm_db module.*



Sufficient filesystem space is available to accommodate the additional tables and new columns added to existing tables in support of the Rumbler. The */db2data/stg0001* and */db2data/stg0002* filesystems should be monitored closely. Record the available free space before starting the Rumbler, and review it again after the first full month of operation. The resulting increase in filesystem usage can be used to estimate how much historical data will impact Db2 storage. Growth will vary depending on the level of transactional activity in the system.

Outline of the setup process is as follows:

1. HPSS Rumbler Package Install
2. Database Preparation
3. Db2 History Table Configuration
4. Rumbler Usage
5. History Table Maintenance

10.2. HPSS Rumbler Package Install

The Rumbler package will be installed via an RPM.

10.3. Database Preparation

10.3.1. Shutdown HPSS and Backup Databases

HPSS must be shut down completely before adding Db2 temporal tables to the system.

```
# /opt/hpss/bin/hpssadm.pl -U hpssssm -a /var/hpss/etc/hpss.unix.keytab
hpssadm> shutdown server -all
```

```
    Shutdown the specified servers? y
    ...
    hpssadm> quit
    # rc.hpss stop
    # su  hpssdb
    $ db2start
```

10.3.2. Database Backups

Generate an offline backup of both databases before modifying the configuration.

```
    $ db2 deactivate db hcfg
    $ db2 deactivate db hsubsys1
    $ db2 backup db HCFG on all dbpartitionnums to <path> compress
    $ db2 backup db HSUBSYS1 on all dbpartitionnums to <path> compress
    $ db2 activate db hcfg
    $ db2 activate db hsubsys1
```

10.4. Db2 History Table Configuration

10.4.1. Add History Tables

The Db2 temporal table definitions are created by the hpss_add_temporal_tables(7) script. The following example assumes the default configuration of a 24-month window (one range partition per month):

```
    $ /opt/hpss/tools/db2_history_prune/package/hpss_add_temporal_tables.py --database
hsubsys1 --months 24 > add_history.log 2>&1
    $ cat add_history.log
    Parsing DDL template file
    Adding History Tablespaces
    Creating History Tables
    Creating History Table Indexes
    Setting History Table Permissions
    Adding Versioning to HPSS Tables that are enabled for History
    Completed adding Db2 history to HPSS database
```

Review the output. Additional details can be obtained by adding the "-v" option, which included the Db2 commands executed to create/modify database objects to support history.

10.4.2. Update Db2 Registry

Verify the db2 registry variables are set to the following:

```
    # su - hpssdb
    $ db2set -all
```

```
[i] DB2FCMCOMM=TCPIP4
[i] DB2COMM=TCPIP
[i] DB2RSHCMD=/usr/bin/ssh
[i] DB2_OVERRIDE_BPF=5000
[g] DB2SYSTEM=<host.domain>
```

There may be additional entries, but ensure that all missing entries are included.

```
$ db2set DB2_OVERRIDE_BPF=5000
```

10.4.3. Update Db2 Timestamp Settings

There are two more changes required to ensure the system is using the correct timestamp information for history operations. First make sure the system is aware of daylight savings with:

```
$ db2 update db config for hsubsys1 using SYSTIME_PERIOD_ADJ YES
```

Second, force the system to utilize UTC reporting. This will need to be updated in the db2profile: `~hpssdb/sqlllib/db2profile`

```
export TZ=UTC
```

10.4.4. Update Db2 Authorization for Access to Stored Procedures and Functions

Db2 stored procedures and functions used by the `hpss_partition_manager.py` tool requires that the *hpss* account be granted authority to access/execute the utilities. That is accomplished with the following as *hpssdb*:

```
$ db2 grant execute on procedure HPSS_UTILS.USP_ADDTABDP to user hpss
$ db2 grant execute on procedure HPSS_UTILS.USP_DETACHTABDP to user hpss
$ db2 grant execute on procedure HPSS_UTILS.USP_DROPPDTAB to user hpss
$ db2 grant execute on function HPSS_UTILS.UDF_GETHPSSDPTABS to user hpss
$ db2 grant execute on function HPSS_UTILS.UDF_GETTABDP to user hpss
$ db2 grant execute on function HPSS_UTILS.UDF_GETIDXDP to user hpss
```

10.4.5. Db2 Runstats and Maintenance

Verify that Db2 Auto Runstats is enabled on hsubsys1. If not set, then enable it.

```
$ db2 get db config for hsubsys1 | grep AUTO_RUNSTATS
     Automatic runstats                  (AUTO_RUNSTATS) = ON
```

Because the history tables are only being appended to, there isn't a need to run the HPSS Db2 maintenance utility on them like the rest of HPSS. The Automatic Runstats is sufficient by Db2. Recommend excluding the Rumbler history tables by adding an exclude list when running weekly Db2 maintenance on the HPSS metadata.

```
$ cat rumbler_exclude_list
BITFILE_HISTORY
NSACL_HISTORY
NSFILESETATTR_HISTORY
NSOBJECT_HISTORY
NSTEXT_HISTORY
NSTRASH_HISTORY
USERATTRS_HISTORY
```

Maintenance would run with:

```
$ Db2_maintenance_utility -d hubsys1 -x rumbler_exclude_list
```

10.4.6. Db2 Backups

Generate new offline backup of databases with the changes.

```
$ db2 deactivate db hcfg
$ db2 deactivate db hsubsys1
$ db2 backup db HCFG on all dbpartitionnums to <path> compress
$ db2 backup db HSUBSYS1 on all dbpartitionnums to <path> compress
$ db2 activate db hcfg
$ db2 activate db hsubsys1
```

10.4.7. Verifying History Setup

Bring Db2 and HPSS fully up. Before doing anything with the system, run the following:

```
$ db2 connect to hsubsys1
$ db2 select "count(*) from hpss.BITFILE_HISTORY"
```

It should report 0 entries from the BITFILE_HISTORY table since nothing has been add/changed.

Now add a file or two into the system. Re-run the select

```
$ db2 select "count(*) from hpss.BITFILE_HISTORY"
```

The select will report the changes to the history table based on the additions, deletions, and removal of files.

10.5. Rumbler Usage

The `rumbler` tool has two modes of operations: Full Run output, Partial Run output. In general, the Full Run is used to obtain a base-line of all files/directories in the HPSS namespace. This can be used to initialize tracking HPSS changes within other systems. The Partial Run output is then used on a periodic basis to update the applications view of files/directories as the namespace changes. Essentially the Partial Run can be used to provide deltas from the last run.



Be sure to recycle HPSS servers before after adding the history table and before running the `rumbler` tool. Otherwise, the following will fail to produce output.

10.5.1. Full Run

The Full Run mode can/will produce a very large output, directly related to size of the HPSS namespace. Be sure the output directory is sufficiently large to store the resulting file. To run:

```
    # /opt/hpss/bin/rumbler -i 1 -u hpssssm -k /var/hpss/etc/hpss.unix.keytab  >
/db2data/db2_backup/full_run.<date>.out
```

The output will look like the following.


```
    {
     "HPSS Subsystem 0001 Buffer 0000000000": [
      {
       "optime": "2024-11-05-15.32.10.268653000000",
       "opendtime": "9999-12-30-00.00.00.000000000000",
       "operation": "U",
       "inode": 1,
       "parentinode": 0,
       "userid": 0,
       "groupid": 0,
       "perms": 16877,
       "type": 132,
       "links": 2,
       "size": 0,
       "blocks": 0,
       "atime": 1730820686,
       "mtime": 1730820680,
       "ctime": 1730820680,
       "birthtime": 1725645734,
       "path": "/"
      },
      {
       "optime": "2024-11-05-15.31.20.828352000000",
       "opendtime": "9999-12-30-00.00.00.000000000000",
       "operation": "U",
       "inode": 3,
       "parentinode": 1,
```

```
      "userid": 312,
      "groupid": 300,
      "perms": 33188,
      "type": 129,
      "links": 1,
      "size": 692460,
      "blocks": 1353,
      "atime": 1730820680,
      "mtime": 1730820681,
      "ctime": 1730820681,
      "birthtime": 1730820680,
      "accountid": 312,
      "cosid": 1,
      "path": "/file.1"
     }
    ]
   }
   {
    "HPSS Subsystem 0001 Buffer 0000000001 Completion Block": [
     {
      "cookie": "2024-11-05-15.32.10.268653000000"
     }
    ]
   }
```


The last part of the file, line containing "cookie", is the ending timeperiod for the run. This is then used as the starting time for the Partial Run.

10.5.2. Partial Run

A Partial Run will use the "cookie" to determine what new entries are to be included in the report.

```
    # /opt/hpss/bin/rumbler -i 1 -u hpssssm -k /var/hpss/etc/hpss.unix.keytab -D 2024-
11-05-15.32.10.268653000000 > /db2data/db2_backup/partial_run.<date>.out
```

That will generate a delta file that records the updates made to HPSS since the last run.


```
   {
    "HPSS Subsystem 0001 Buffer 0000000000": [
     {
      "optime": "2024-11-05-15.31.10.274088000000",
      "opendtime": "2024-11-05-15.32.10.268653000000",
      "operation": "U",
      "inode": 1,
      "parentinode": 0,
      "userid": 0,
      "groupid": 0,
      "perms": 16877,
```

```
        "type": 132,
        "links": 2,
        "size": 0,
        "blocks": 0,
        "atime": 1730820623,
        "mtime": 1725645734,
        "ctime": 1725645734,
        "birthtime": 1725645734,
        "path": "/"
       },
       {
        "optime": "2024-11-05-18.15.28.688592000000",
        "opendtime": "2024-11-05-18.15.29.688706000000",
        "operation": "U",
        "inode": 4,
        "parentinode": 1,
        "userid": 312,
        "groupid": 300,
        "perms": 33188,
        "type": 129,
        "links": 1,
        "size": 0,
        "blocks": 0,
        "atime": 1730830528,
        "mtime": 1730830528,
        "ctime": 1730830528,
        "birthtime": 1730830528,
        "accountid": 312,
        "cosid": 1,
        "path": "/file.2"
       },
       {
        "optime": "2024-11-05-18.15.29.710233000000",
        "opendtime": "9999-12-30-00.00.00.000000000000",
        "operation": "U",
        "inode": 4,
        "parentinode": 1,
        "userid": 312,
        "groupid": 300,
        "perms": 33188,
        "type": 129,
        "links": 1,
        "size": 692460,
        "blocks": 1353,
        "atime": 1730830528,
        "mtime": 1730830529,
        "ctime": 1730830529,
        "birthtime": 1730830528,
        "accountid": 312,
        "cosid": 1,
        "path": "/file.2"
```

```
      },
      {
       "optime": "2024-11-05-18.15.37.599877000000",
       "opendtime": "9999-12-30-00.00.00.000000000000",
       "operation": "U",
       "inode": 1,
       "parentinode": 0,
       "userid": 0,
       "groupid": 0,
       "perms": 16877,
       "type": 132,
       "links": 2,
       "size": 0,
       "blocks": 0,
       "atime": 1730820686,
       "mtime": 1730830528,
       "ctime": 1730830528,
       "birthtime": 1725645734,
       "path": "/"
      }
     ]
    }
    {
     "HPSS Subsystem 0001 Buffer 0000000001 Completion Block": [
      {
       "cookie": "2024-11-05-18.15.37.599877000000"
      }
     ]
    }
```


And the "cookie" output from this run will be used for the next Partial Run starting timestamp.

10.6. History Table Maintenance

10.6.1. Overview

History tables require routine maintenance to keep HPSS/Db2 functioning properly. Every time an HPSS file or directory is modified or removed, one or more associated history records are generated resulting in additional metadata. Part of the maintenance is to create additional history table range partitions (not to be confused with database partitioning) for future months. As a starting point, the number of history table range partitions is defined during setup with the hpss_add_temporal_tables(7) tool. The default is 24 months (one range partition per month) which should be adequate buffer for most sites.



As time progresses towards the two-year mark, additional monthly range partitions must be created or HPSS functionality will be impacted. History table entries become part of the entire transaction operation (i.e., creating an HPSS file), and if it cannot be completed, then the entire transaction is rolled back. Do not wait until the end of

> *the last range partition to add space.*

While new history table range partitions are created for future months, older table partitions will need to be pruned (dropped). They are no longer required by the Rumbler after reports covering that period have been acquired. Consequentially, they just take up storage. The current recommendation is to keep 2 months' worth of older history table range partitions. After that, the older range partitions should be removed and storage freed. This section discusses in detail the maintenance procedure and tools to aid in this task.

10.6.2. Additional Setup for Manage Partitioning Tool

Python requires additional modules to support the Rumbler. Some of these may already exist on the system. If so, those specific items may be skipped.

10.6.2.1. ibm_db and pycrypto modules

Make sure *ibm_db* and *pycrypto* are installed for Python for the hpssdb account.

```
    $ pip show ibm_db
    $ pip show pycrypto
```

If either is missing, then install.

10.6.2.2. HPSS Client *hpss* module

The *hpss* module should be installed when the *hpss-clnt* RPM is installed. However, we have noticed that this is sometimes not the case. In that event, as root, perform the following steps:

```
# cd /opt/hpss/clapi
# python setup.py install
```

You can verify the module is in place by importing *hpss* like the following:

```
    # python
    Python 3.11.14 (main, Oct 22 2025, 16:52:56) [GCC 11.3.1 20221121 (Red Hat 11.3.1-
4)] on linux
    Type "help", "copyright", "credits" or "license" for more information.
    >>> import hpss
    >>>
```

The return of the ">>>" prompt without an error verifies the import was successful.

10.6.2.3. Edit the udf.conf file

Create or edit the `/var/hpss/etc/udf.conf` file. The contents will look similar to:

```
    DEBUG = ON
    SOURCE_ENDIANNESS = LITTLE_ENDIAN
    TARGET_ENDIANNESS = LITTLE_ENDIAN
    SYSTEM_ID         = XXXXX          <-------- **Replace**, system dependent
    HASH_SEED         = YYYYY          <-------- **Replace**, system dependent
    OUTPUT_FILE = /tmp/UDFSleep.log
```

The **XXXXX** and **YYYYY** values must be replaced with values from the current system. These can be obtain using the lshpss -global command:

```
    # /opt/hpss/bin/lshpss -global

    ...
    System ID:              1
    Hash Algorithm Seed:        1022608322
    ...
```

Now run the `udfcreate.sh` tool

```
    # /opt/hpss/tools/metadata/udf/udfcreate.sh
```

It should report several lines consisting of `"DB20000I The SQL command completed successfully."` messages.

10.6.3. Configuration Files

Copy the following files to /var/hpss/etc:

```
    # cp /opt/hpss/tools/db2_history_prune/package/manage_temporal.conf
/var/hpss/etc/manage_temporal.conf
    # cp /opt/hpss/tools/db2_history_prune/package/log_config.yaml
/var/hpss/etc/manage_temporal.yaml
    # cp /opt/hpss/tools/db2_history_prune/package/db.conf
/var/hpss/etc/rumbler.db.conf
```

Edit the `manage_temporal.conf` file, change the *admin_file* line to reflect new path/name of file:

```
    admin_file = /var/hpss/etc/rumbler.db.conf
```

Create the logging directory for the tools:

```
    # mkdir /var/hpss/log/manage_temporal
    # chown hpssdb.hpss /var/hpss/log/manage_temporal
```

If another directory is to be used for this output, the `/var/hpss/etc/manage_temporal.yaml` file will require editing to reference the new path.

Create the file `/var/hpss/etc/hpss_partition_manager.conf` with the following content:

```
    [DEFAULT]
       tables =
NSACL_HISTORY,NSFILESETATTR_HISTORY,NSTEXT_HISTORY,NSTRASH_HISTORY,USERATTRS_HISTORY,N
SOBJECT_HISTORY,BITFILE_HISTORY
```

Earlier versions of the Rumbler package will need the following symbolic links. (The RPM version of the package with 12.1 will reference `/var/hpss/etc` instead of `/etc`.)

```
    # ln -s /var/hpss/etc/manage_temporal.conf /etc/manage_temporal.conf
    # ln -s /var/hpss/etc/manage_temporal.yaml /etc/manage_temporal.yaml
    # ln -s /var/hpss/etc/hpss_partition_manager.conf /etc/hpss_partition_manager.conf
```

10.6.4. History Management Tools

There are 2 history management tools provided in `/opt/hpss/tools/db2_history_prune/package` directory

- manage_temporal.py – provides ability to list, add new partitions, and remove old ones, for a specific history table.
- hpss_partition_manager.py – provides ability to list, add new partitions and remove old ones, for all history tables simultaneously.

It is recommended to use the `hpss_partition_manager.py` tool. The tool is easier to use, and helps the admin maintain all the Rumbler history tables in one operation. Only the hpss_partition_manager.py is used in the examples below.

Refer to the README file in the directory `/opt/hpss/tools/db2_history_prune` for more information.

10.7. Viewing History Table Partitions

Execute `hpss_partition_manager.py` using the *hpssdb* account:

```
    $ /opt/hpss/tools/db2_history_prune/package/hpss_partition_manager.py -a list
    Instance: hpssdb Database: hsubsys1 Schema: hpss Table: NSACL_HISTORY Partition:
PART0 Low: 2024-11-01 00:00:00 High:   2024-12-01 00:00:00
    Instance: hpssdb Database: hsubsys1 Schema: hpss Table: NSACL_HISTORY Partition:
PART1 Low: 2024-12-01 00:00:00 High:   2025-01-01 00:00:00
    Instance: hpssdb Database: hsubsys1 Schema: hpss Table: NSACL_HISTORY Partition:
PART2 Low: 2025-01-01 00:00:00 High:   2025-02-01 00:00:00

    ...
```

```
    Instance: hpssdb Database: hsubsys1 Schema: hpss Table: NSACL_HISTORY Partition:
PART23 Low: 2026-10-01 00:00:00 High:  2026-11-01 00:00:00
    Instance: hpssdb Database: hsubsys1 Schema: hpss Table: NSACL_HISTORY Partition:
PART24 Low: 2026-11-01 00:00:00 High: 2026-12-01 00:00:00

    Instance: hpssdb Database: hsubsys1 Schema: hpss Table: NSFILESETATTR_HISTORY
Partition: PART0 Low: 2024-11-01 00:00:00 High: 2024-12-01 00:00:00
    Instance: hpssdb Database: hsubsys1 Schema: hpss Table: NSFILESETATTR_HISTORY
Partition: PART1 Low: 2024-12-01 00:00:00 High: 2025-01-01 00:00:00

    ...
```

The listing continues, reporting the last partition of the last history table (from the `/var/hpss/etc/hpss_partition_manager.conf` configuration file)

If you want to know how many data pages are consumed by each history partition you can run:

```
    $ db2 connect to hsubsys1
    $ db2 "select npages as Number_of_pages,SUBSTR(LOWVALUE,2,10) AS
Start_Date,SUBSTR(HIGHVALUE,2,10) AS End_Date from  SYSCAT.DATAPARTITIONS WHERE
TABSCHEMA = 'HPSS' and TABNAME='NSOBJECT_HISTORY'"

    NUMBER_OF_PAGES      START_DATE END_DATE
    -------------------- ---------- ----------
                       1 2024-11-01 2024-12-01
                       0 2024-12-01 2025-01-01
                       0 2025-01-01 2025-02-01
                       0 2025-02-01 2025-03-01
                       0 2025-03-01 2025-04-01
                       0 2025-04-01 2025-05-01
                       0 2025-05-01 2025-06-01
                       0 2025-06-01 2025-07-01
                       0 2025-07-01 2025-08-01
                       0 2025-08-01 2025-09-01
                       0 2025-09-01 2025-10-01
                       0 2025-10-01 2025-11-01
                       0 2025-11-01 2025-12-01
                       0 2025-12-01 2026-01-01
                       0 2026-01-01 2026-02-01
                       0 2026-02-01 2026-03-01
                       0 2026-03-01 2026-04-01
                       0 2026-04-01 2026-05-01
                       0 2026-05-01 2026-06-01
                       0 2026-06-01 2026-07-01
                       0 2026-07-01 2026-08-01
                       0 2026-08-01 2026-09-01
                       0 2026-09-01 2026-10-01
                       0 2026-10-01 2026-11-01
                       0 2026-11-01 2026-12-01
```

Note that future tables don't take up space; one reason they can be safely created ahead of time.

If you just want to know the total number of pages used and the date ranges allocated, run:

```
    db2 "select sum(npages) as Total_pages,min(SUBSTR(LOWVALUE,2,10)) AS
Start_Date,max(SUBSTR(HIGHVALUE,2,10)) AS End_Date from  SYSCAT.DATAPARTITIONS WHERE
TABSCHEMA = 'HPSS' and TABNAME='NSOBJECT_HISTORY'"

    TOTAL_PAGES          START_DATE END_DATE
    -------------------- ---------- ----------
                       1 2024-11-01 2026-12-01
```

10.7.1. Table Pruning

As time progresses, the older history table range partitions are no longer needed (for now, plan to keep at most 2 full months of older ranges). Example: its November, keep range partitions for September and October, the range partitions for August and earlier are no longer relevant assuming the Rumbler has been executed over those time periods. These range partitions can be removed using the `hpss_partition_manager.py` tool. After dropping these range partitions, the space is returned to the database and available for use. Syntax for removal is:

```
    $ /opt/hpss/tools/db2_history_prune/hpss_partition_manager.py -a remove -M <nn>

    # Where <nn> is the number of range partitions prior to the current range
partition, for which we want to KEEP. The default is 1.
```

For example:

```
    $ /opt/hpss/tools/db2_history_prune/hpss_partition_manager.py -a remove -M 2
```

This would keep history table range partitions for the current month, and the previous two months for all tables associated with Rumbler operations. The "-t" option can be used to limit operations to a specific table.

10.7.2. Adding Partitions to Tables

You should make sure that all history tables have partitions covering the current month and at least 2 additional months after that. Note that empty partitions do not use storage space.



As time progresses towards the two-year mark, additional monthly range partitions must be created or HPSS functionality will be impacted. History table entries become part of the entire transaction operation (i.e., creating an HPSS file), and if it cannot be completed, then the entire transaction is rolled back. Do not wait until the end of the last range partition to add space.

Syntax for adding partitions is:

```
    $ /opt/hpss/tools/db2_history_prune/package/hpss_partition_manager.py -a add -M
<nn>

    # Where <nn> is the number of months in the future for which we want partitions
(default is 1)
```

Example, the number of additional months already configured for partition in 6. We want the extend the number of partitions by 3 additional months. We would issue the following:

```
    $ /opt/hpss/tools/db2_history_prune/package/hpss_partition_manager.py -a add -M 9
```



Note that it's not 9 months added to the existing 6 but rather extending the total number of months to 9.

Chapter 11. HPSS S3 interface

HPSS offers an S3 (Simple Storage Service) interface. The S3 interface is a popular object storage interface and has been popularized by Amazon Web Services (AWS) S3. This document will describe the HPSS S3 interface, the goals behind the interface, its limitations, different options, deployment considerations, and troubleshooting and debugging tips.

Note that while the HPSS S3 interface provides functionality similar to the AWS S3 interface, specific aspects of how operations work may differ. This document is meant to describe the setup and usage of the HPSS S3 interface. AWS S3 documentation may serve as a guide for what to expect from the HPSS S3 interface, but anything specific to HPSS will supersede guidance from other sources.

That said, AWS provides a number of resources for understanding S3 at a high level, and provides documentation on programming against an S3 endpoint.

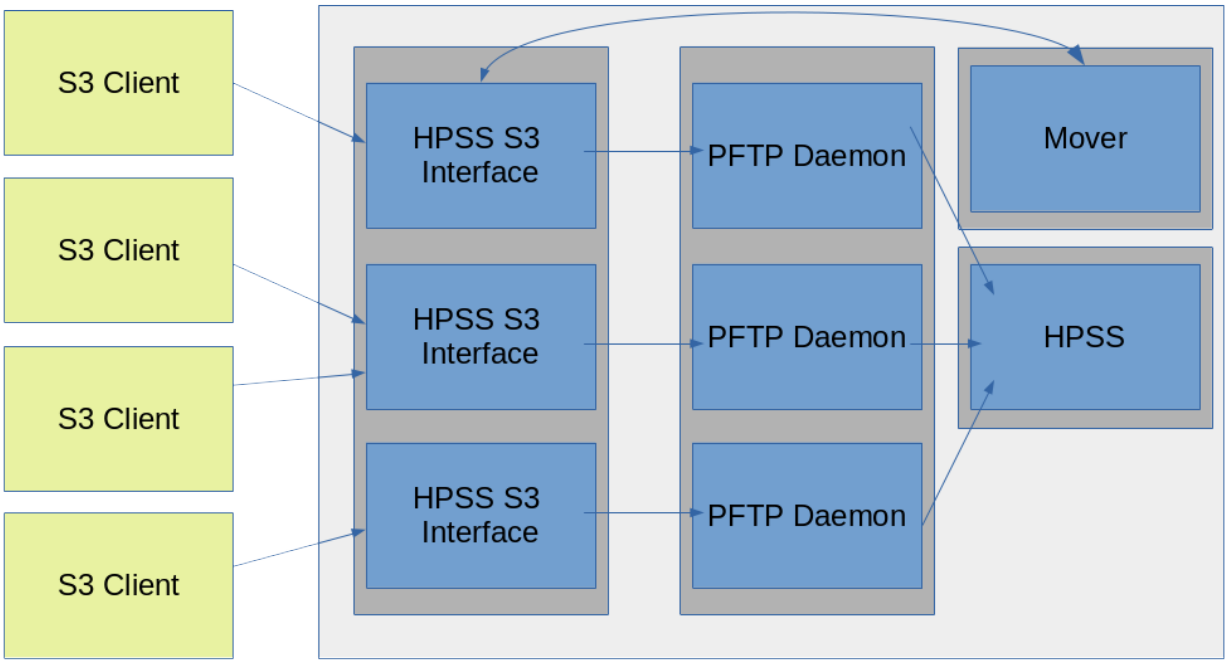
11.1. Overview

The HPSS S3 interface is an HTTP server that processes S3 REST API requests and forwards them to HPSS. The interface uses the HPSS PFTP server as its primary means of communicating with HPSS. The HPSS PFTP server provides mechanisms for using HPSS features through an FTP interface such as COS selection, file family selection, file hashing, user-defined attributes, and high-speed data transfers.

The HPSS S3 interface allows users to list, get, put, and remove HPSS files. The S3 specification has its own limitations, some of which are lower than HPSS limits. When a conflict exists between the two, the S3 limit is enforced.

A key factor of the HPSS S3 interface is that it allows for seamless integration of existing data and user interfaces and new data coming in through S3. Files already in HPSS, or added through other interfaces such as HPSSFS or HSI, are accessible through the HPSS S3 interface if that user would have read access to those files normally. This is explained in detail in the Interoperation section below.

The HPSS S3 interface should be deployed on a client system with sufficient CPU to manage validating parallel checksum streams. Multiple HPSS S3 interfaces can be deployed on the same client system. Once deployed, S3 clients can contact the HPSS S3 HTTP servers and issue their S3 commands with minimal setup.



Note that the HPSS S3 interface is **compatible** with S3, rather than **compliant**. That is, there are some aspects of S3 which the HPSS S3 interface does not support and are not implemented. These are described in detail in Unsupported Operations.

HPSS S3 ultimately logs into HPSS using an HPSS user's username and password. S3 authentication uses ACCESS KEY and SECRET KEY for authentication. The ACCESS KEY will be the user's HPSS accounts to log in. The SECRET KEY will be generated by the administrator and used to map to a user's credentials on the server side.

11.1.1. End User Authentication

HPSS S3 supports end user authentication via both a password database and temporary credentials provided using Open ID Connect authentication services.

11.1.1.1. Password Database

To utilize authentication via password database, a java keystore will need to be generated that holds the ACCESS KEY as the alias and the SECRET KEY as a password. The java keystore will be generated as follows:

```
keytool -importpass -storetype pkcs12 -alias <hpss user> -keystore <hpss keystore
path>
```

Add the keystore path and keystore password to hpss_s3proxy.conf as s3proxy.hpss-creds and s3proxy.hpss-ks-password respectively.

For example:

```
root@tidus /opt/hpss > keytool -importpass -storetype pkcs12 -alias hpss -keystore
```

```
/var/hpss/etc/hpsss3keystore
Enter keystore password:  hpsshpss
Re-enter new password:  hpsshpss
Enter the password to be stored: hpsspassword
Re-enter password: hpsspassword
```

For this example, you would add the following properties to hpss_s3proxy.conf:

- s3proxy.hpss-creds = /var/hpss/etc/hpsss3keystore
- s3proxy.hpss-ks-password = hpsshpss

11.2. HPSS S3 Interface Setup

To set up the S3 interface, first set up PFTP on the client system. This is covered in the HPSS User's Guide. PFTP is used as the backend application to support S3 operations. The PFTP client should be set up to allow any users that you plan to use the S3 interface to authenticate through the PFTP client.

Next, obtain the HPSS S3 proxy binary. This is a Java binary, so a Java runtime must be installed on the system. See the HPSS Admin Guide for Java runtime version requirements. There are no further software requirements to run the HPSS S3 proxy.

Determine what HPSS COS will be used for writing files. A specific HPSS COS can be specified below in the configuration file or it can be left blank or set to 0 for FTP to decide based on file size. On the server identified by jclouds.endpoint, configure the PFTP Daemon in the /var/hpss/etc/HPSS.conf file in the following section:

```
        # Uncomment next line to use the COS from the FileSize Options
        # The default is to ignore the COS specification in the Table.
        ; Set COS Based on Filesize

        # Specify Blocksizes, StripeWidths, and COSs to use based on file size
        # The table must be in strictly ascending order of filesize.
        #
        # COS has no meaning to the Non-HPSS PFTP Daemon
        # COS = 0 means allow the BitFile Server to determine the optimal COS
        #
        # BlockSize has no meaning to the HPSS PFTP Daemon
        #
        # StripeWidth = 0 means use the Bitfile Server Value (HPSS PFTP Daemon)
        #         or use the default (Non-HPSS PFTP Daemon)
        ; FileSize Options = {
                # Files greater than or equal to this value and less than
                # any other value in the table use these Settings
                # e.g., 2MB <= filesize < 10MB
                ; 1MB = {
                        ; BlockSize = 512KB
                        ; StripeWidth = 0
                        ; COS = 2
```

```
                ; }
                ; 2MB = {
                        ; BlockSize = 2MB
                        ; StripeWidth = 4
                        ; COS = 0
                ; }
                ; 10MB = {
                        ; BlockSize = 2MB
                        ; StripeWidth= 0
                        ; COS = 0
                ; }
                ; 100MB = {
                        ; BlockSize = 4MB
                        ; StripeWidth= 0
                        ; COS = 0
                ; }
                ; 1GB = {
                        ; BlockSize = 8MB
                        ; StripeWidth= 0
                        ; COS = 0
                ; }
        ; }
```

Determine or create a directory for the S3 buckets to be created in. This can either be an existing directory within HPSS or a new directory. This will become the base directory listed below in the configuration file.

Configure the hpss_s3.conf file.

Configuration Item	Description	Default
s3proxy.authorization	S3 Authorization Type	AWS_V2_or_V4
s3proxy.kerberos.realm	Kerberos realm	<none>
s3proxy.kerberos.ktname	Keytab with HTTP cred	<none>
s3proxy.endpoint	Endpoint URL	URL:PORT
s3proxy.secure-endpoint	Secure endpoint URL	URL:PORT
s3proxy.keystore-path	SSL keystore path	<none>
s3proxy.keystore-password	SSL keystore password	<none>
s3proxy.hpss-creds	Credential keystore	/var/hpss/etc/hpsss3keystore
s3proxy.hpss-ks-password	Keystore password	hpsshpss
s3proxy.header-cache-size	HTTP Header Cache Size	32768
s3proxy.request-header-size	HTTP Request Header Size	32768
s3proxy.response-header-size	HTTP Response Header Size	32768
s3proxy.output-buffer-size	HTTP Output Buffer Size	524288

s3proxy.transfer-buffer-size	HTTP Transfer Buffer Size	524288
s3proxy.hpss-auto-stage	Disable Automatic Stage From Tape	true
s3proxy.v4-max-non-chunked-request-size	Largest allowed non-chunked put request	33554432
s3proxy.virtual-host	Virtual hosted URL support	<none>
s3proxy.region	AWS-like Region Name	us-east-2
s3proxy.authtkn.endpoint	Authentication Service endpoint URL	<none>
s3proxy.authtkn.keystore	Authentication Service credential keystore	/var/hpss/etc/s3proxy_authtkn.jks
s3proxy.authtkn.keystore-password	Authentication Service credential keystore password	<none>
s3proxy.authtkn.oidc-provider-url	URL for the authorization service	<none>
s3proxy.authtkn.realm-name	Authentication Service security realm name	hpss_s3_realm
s3proxy.authtkn.client-name	Authorization service client identifier	s3proxy-client
s3proxy.authtkn.client-secret	Authorization service client secret	<none>
s3proxy.authtkn.remote	Do all session verification as if the authentication service was remote **(DO NOT USE IN PRODUCTION)**	false
s3proxy.authtkn.signing-key-lifetime-hours	Number of hours before changing the token signing key	24
s3proxy.authtkn.token-lifetime-hours	Number of hours an authentication session will remain valid	720
s3proxy.authtkn.encryption-key	32 character key used for encryption of the secrets maintained on local disk	<none>
s3proxy.authtkn.remote-signing-key	32 character key used to sign token decode requests.	<none>
s3proxy.authtkn.active-token-store-path	Path on local disk to maintain database of active sessions	/var/hpss/etc
s3proxy.authtkn.session-eviction-multicast-ip	Multicast IP address used to communicate session revocation	232.30.30.10

s3proxy.authtkn.session-eviction-multicast-port	Multicast IP port used to communicate session revocation	4446
s3proxy.authtkn.hpss-username-claim	Id token claim name that identifies the associated HPSS username	preferred_username
s3proxy.authtkn.session-adm-role-claim	Id token claim used to infer S3 Proxy roles	roles
s3proxy.authtkn.session-adm-role-adm	Name of the administrative role identifier	s3proxy_admin
s3proxy.authtkn.session-adm-role-opr	Name of the operator role identifier	s3proxy_operator
jclouds.pftp.hpss.cos	Default HPSS COS	<Default COS>
jclouds.pftp.hpss.swidth	Default HPSS client stripe transfer width (range 1-32)	1
jclouds.ftp.basedir	Base Directory	<HPSS Mount>
jclouds.endpoint	Endpoint	PFTP Server:Port
jclouds.pftp.hpss.max.connections	Max PFTP Connections	Must match xinetd limits
jclouds.pftp.multipart.upload.connections	Default number of HPSS file opens per multipart upload	1
jclouds.pftp.multipart.upload.connections.limit	Maximum number of HPSS file opens per multipart upload	6
jclouds.pftp.dirent.cache.dir	Directory used for caching object listing results	/var/hpss/tmp
jclouds.pftp.dirent.cache.overflow	Number of object listing entries to keep in memory	1000
jclouds.pftp.dirent.cache.timeout	Number of seconds to maintain object listing results	300
jclouds.pftp.dirent.list.timeout	Number of seconds to wait for HPSS to respond with listing results	60
hpsspftp.sockbuf.send.size	HPSS PFTP Send Socket Buffer	524288
hpsspftp.sockbuf.recv.size	HPSS PFTP Recv Socket Buffer	524288
hpsspftp.iobuf.size	HPSS PFTP I/O Buffer Size	524288
hpsspftp.s3tkn.privacy	Enable PFTP secure channel privacy for S3 session token log in	false

Set up the service file in systemd and start the service. Note that multiple services can be

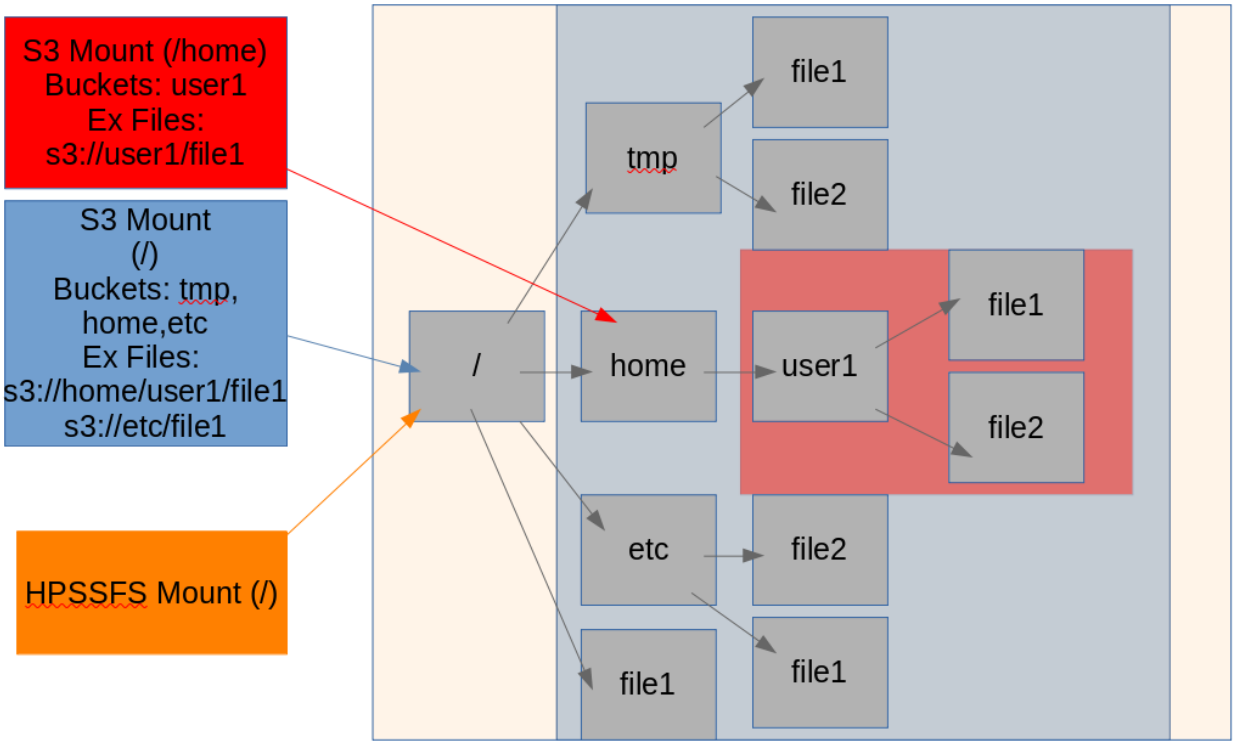
configured for different servers with different configurations, with minor tweaks to the provided service file.

```
# cp hpss_s3proxy.service /etc/systemd/system/
# systemctl daemon-reload
# systemctl enable hpss_s3proxy
# systemctl start hpss_s3proxy
# systemctl status hpss_s3proxy
```

Additionally, users allowed to authenticate to the S3 interface must be defined. Secret keys must be generated by the administrator and provided to users. A mapping file will map HPSS user IDs to secret keys for use in validating incoming requests.

11.3. Interoperation

The HPSS S3 interface is able to interoperate with existing HPSS files and workflows from other HPSS interfaces such as HPSSFS, HTAR, HSI, and PFTP.



A user who interacts with the HPSS S3 interface will see the directories in the basedir as S3 buckets. Everything underneath those buckets are considered objects to the S3 interface. Consider, for example, a path such as /home/user1/file1, where the S3 interface is mounted to /home. user1 is a treated as a bucket in this case, and the object path is "file1" inside the user1 bucket. To reference the same file with the S3 interface mounted on /, "home" is the bucket and the object name is "user1/file1". In the S3 context, "user1/" in this path is referred to as a prefix. When listing buckets which contain multiple subdirectories with files, it is useful to filter based on prefixes or use options to limit recursion.

The S3 interface mounted on /home cannot reference /tmp, /etc/ or /file1. The S3 interface mounted on / also cannot reference /file1 because it is not in a directory (bucket).

The S3 interface is not sequestered from the HPSS global namespace. The basedir configured for the mount point will point at a real HPSS directory, whether it be the root or some other directory such as a home directory. When a user lists the buckets, they will see each directory under the base directory that they have read access to. They will not see directories they do not have read access to. Similarly, when a user lists objects within a bucket, they will see all objects they have read access to, and no objects they do not have read access to.



On bucket deletions, the S3 interface will verify that the requesting user is the owner of the specified bucket. However, for an added measure of protection it is advised that the sticky bit be set on the basedir to prevent inadvertent bucket deletion outside the S3 interface.

The HPSS S3 interface does minimal local caching so objects created or deleted by other users in other applications should behave as expected for S3 users. Multi-user access across multiple S3 interfaces and applications has the same potential dangers for overwrites as other interfaces would, so care should be taken by users to avoid overwriting the same file in parallel.

S3 has a concept of "Storage Classes". This has no relation to HPSS Storage Classes. The S3 Storage Classes describe an access pattern or expected latency for retrieval (see the AWS S3 documentation for details). Within the HPSS S3 Interface, S3 objects will be designated with an S3 Storage Class of **INTELLIGENT_TIERING**. Any storage class explicitly specified when the object was uploaded will be saved and made available for query via the *x-amz-meta-x-s3proxy-meta-storage-class* attribute .

Direct accessibility of HPSS S3 objects may be determined by either trying to get the object, or by a query of the object attributes. Get requests for archived objects will fail with an HTTP "Bad Request" status and a reason code of "Invalid Object State". The archive status of an object can also be determined by the presence of the *x-amz-archive-status* attribute. If this attribute is not returned, then the object is ready for immediate access. If the attribute is returned the associated value will be **ARCHIVE_ACCESS**, and the object must first be restored prior to access.

11.4. HPSS Specific Options

HPSS supports a set of custom headers used to modify the behavior of HPSS. Most clients allow custom headers to be sent along with S3 requests.

x-amz-meta-hpss-cos

HPSS Class of Service to use. This only impacts S3 put operations.

x-amz-meta-hpss-family

HPSS File Family to use. This only impacts S3 put operations.

x-amz-meta-hpss-swidth

HPSS client stripe transfer width to use. Client stripe transfers leverage multiple TCP/IP connections to improve performance for non-striped volume I/O.

x-amz-meta-hpss-mpp-length

HPSS file length hint for multipart uploads. This only impacts S3 multipart upload operations for fixed segment HPSS classes of service. See information for the **Allocation Method** parameter on

the Class of Service Configuration Window for more details.

HPSS supports trashcans. When trashcans are enabled, files deleted within the HPSS S3 interface will be placed in the trashcan. The trashcan may or may not be accessible to end users depending upon the base directory specified.

11.5. S3 Multipart Upload and HPSS File Handles

Multipart uploads can take advantage of one or more HPSS file handles based on the configuration items. The *jclouds.pftp.multipart.upload.connections* configuration item indicates the default number of HPSS file handles to use per multipart upload request. The *jclouds.pftp.multipart.upload.connections.limit* configuration variable indicates the limit on number of open HPSS handles used per multipart upload requests. The default number of handles is always maintained unless a part upload request is in danger of exceeding the idle timeout, specified via jetty.http.IdleTimeout (default 30 seconds). After this point a new HPSS file handle will be opened while the current number of open handles remains under the specified limit.

On multipart upload a multipart etag will be generated and can be accessed via the *x-amz-meta-pftp-multipart-upload-etag* attribute.

11.6. S3 Object Listing

To support the timely listing of larger name spaces the S3 proxy will temporarily cache object listing results in memory and on disk, allowing the client to retrieve partial listing results without incurring extreme latencies. The *jclouds.pftp.dirent.** parameters control the behavior of this capability. The *jclouds.pftp.dirent.cache.overflow* parameter controls the number of object listing entries that will be maintained in memory. Object listing entries over this number will be stored in a temporary file on disk. These temporary files have the following naming format: *s3proxy-dirent-<unique identifier>.lst*. The *jclouds.pftp.dirent.cache.dir* parameter is used to control the directory where the temporary files are stored. The *jclouds.pftp.dirent.timeout* parameter controls the number of seconds of inactivity that must pass before an idle list cache becomes a candidate for removal.

11.7. S3 Virtual Hosting

For virtual hosted style URLs, the bucket name is part of the domain name supplied in the request. In order to support these types of URLs, the *s3proxy.virtual-host* configuration item is used to indicate what portion of the URL is the domain name and what portion is the bucket name. For example with *s3proxy.virutal-host* set to **myhost.mycompany.com**, an URL specifying **mybucket.myhost.mycompany.com** would indicate that the request be directed to objects in the **mybucket** bucket.

11.8. S3 Buckets

Buckets visible via your S3 proxy instance correspond to the directories present in the HPSS base directory that conform to the S3 proxy bucket naming rules. The HPSS base directory is specified using the *jclouds.ftp.basedir* configuration variable. Attempts to create or access buckets that do not

conform to the bucket naming rules will result in failure.

The following naming rules apply for S3 proxy buckets.

- Bucket names must be between 3 (min) and 255 (max) characters long.
- Bucket names can consist only of uppercase letters, lowercase letters, numbers, dots (.), underscores (_), and hyphens (-).
- Bucket names must begin and end with a letter or number.
- Bucket names must not end with dots (.).
- Bucket names must not be formatted as an IP address (for example, 192.168.5.4).

11.9. S3 Client Setup

Different S3 clients will have different setup requirements. Consult client-specific documentation for general setup instructions and best practices. Below are some general tips and some examples for using the HPSS S3 with clients.

11.9.1. General

The general items that need to be configured for any S3 client will be:

1. **The S3 endpoint.** This is the URL that the client will attempt to contact for S3 services. This is typically a HTTP URL with a port number.
2. **The Access Key.** This is the access key used to authenticate to S3. This will correspond to the HPSS user name.
3. **The Secret Key.** This is the secret key used to sign requests to the S3 requests to the server. The server will verify the request using this secret key, provided to the user.
4. **Use Path Bucket URLs.** This is an older method for specifying bucket URLs, but it is widely supported across clients. Path bucket URLs are URLs where the bucket path is appended to the URL endpoint (e.g. example.com/tmp) and is distinguished from sub-domain URLs (e.g. tmp.example.com). The HPSS S3 interface only supports path bucket URLs.

Some clients may also require specifying the signature version. This is the method used to sign requests to the server with the secret key. AWS V4 (also called S3v4) signatures are the current standard. AWS V2 was the prior standard which has been removed. The HPSS S3 interface only supports the AWS v4 signature.

11.9.2. s3cmd

For s3cmd, in the `s3cmd.cfg` file, set the `access_key`, the `host_base`, the `host_bucket`, and the `secret_key`. The `send_chunk` and `recv_chunk` may be modified to tune performance. Additionally, the `host_bucket` should be set to use a post-fix bucket in the form `<ip>:<port>/(%bucket)`.

For example:

```
host_bucket = 127.0.0.1:8080/(%bucket)
```

s3cmd supports the `--add-header` option, which can be used to specify certain HPSS Specific Options.

11.9.3. boto3

For boto3, the endpoint_url, aws_access_key_id, and aws_secret_access_key must be provided for s3 clients and resources.

boto3 does not support custom headers directly, but headers can be injected by using a simple snippet like:

Boto3 client example

```
def _add_header(request, **kwargs):
    request.headers.add_header('x-amz-meta-hpss-cos', '1')

s3 = boto3.client(...)

event_system = s3.meta.events
event_system.register_first('before-sign.*.*', _add_header)
```

11.10. Unsupported Operations

In general, the HPSS S3 interface supports basic operations like make bucket, list bucket, delete bucket, put object, get object, and delete object. A lot of additional options have been added to S3 over time. Some of these options only make sense in a larger AWS context, some are not currently implemented but may be in the future.

The following S3 features are unsupported at this time:

- CORS
- Object Versioning
- Bucket / Object encryption
- Intelligent Tiering
- Inventory Configuration
- Bucket Lifecycle
- Bucket Metrics
- Bucket Policies
- Bucket Replication
- Bucket Tagging
- Object Tagging
- Bucket Website Configuration

- Bucket Transfer Acceleration
- Bucket Analytics Configuration
- Bucket Logging
- Bucket Notifications
- Bucket Request Payment
- Object Legal Hold
- Object Retention
- Object Torrent
- Select Object content
- Object Lambda access points

These features may have fields in otherwise supported operations which are not implemented or supported. For example **list object parts** allows parameters like RequestPayer (not supported) to specify a payer and SSECustomerKey (not supported) to specify an encryption key. While **list object parts** is supported, these additional parameters related to unsupported features are not.

11.11. Limitations

- In the effort to maintain consistency between S3 and file system clients, S3 key components (delimited by forward slashes) are represented in HPSS as directories. This can place limitations on the how key names can be generated. For example the following key names would be perfectly allowable in an AWS S3 bucket.

  ```
     North-American
     North-American/Eastern
  ```

  However these keys would be forbidden in an HPSS S3 bucket, because "North-American" would be created as an HPSS file. The subsequent put of "North-American/Eastern" would fail because "North-American" is not a directory. Failures due to this limitation should result in an HTTP reply status indicating that the behavior is not implemented.

  ```
  Example:
  ```

  ```
     Code 501 - 'NotImplemented'
     Message:'Component of this path already exists as an object: North-
  American/Eastern'
  ```

- Currently, key names must not include line returns. Requests utilizing keys with line returns will fail with reply status indicting that the behavior is not implemented.

```
For example, key North-American/Eastern<line return> would result in:
```

```
   Code 501 - 'NotImplemented'
   Message:'Path contains unsupported characters: North-American/Eastern
'
```

11.12. Performance

The first place to start tuning HPSS S3 is with PFTP. Verify that a PFTP client on the system where the HPSS S3 interface is deployed can achieve the expected transfer rates. Tune the PFTP client settings in HPSS.conf to achieve the expected rates based on the network and underlying HPSS hardware.

Once the PFTP interface is well tuned, it is time to tune the S3 interface. The HPSS S3 interface uses a high-performance jclouds module which interacts with PFTP as a PFTP client. This enables the HPSS S3 interface to achieve high performance throughput without a "store and forward" strategy of some other S3 interfaces. Additionally, the S3 interface makes use of a high performance md5 checksumming module to generate eTags, which allows PUT and GET operations to achieve hundreds of MB/s of performance with a minimal amount of CPU overhead.

Some client applications will generate their own checksum to send to the S3 server to verify. This can be time-intensive and is a data integrity vs performance trade-off if it can be turned off. The S3 interface must calculate and return an ETag which in the case of the HPSS S3 interface is MD5.

Client applications also have their own network parameters that may need to be tuned. The HPSS S3 interface usually benefits from a larger socket and buffer size.

11.13. Scaling and Load Balancing

The HPSS S3 interface can be placed behind a load balancing solution such as haproxy for redundancy, scalability, and performance. Multiple HPSS S3 instances can be placed on the same system, or can be spread across multiple client systems. The PFTP layer can be scaled out to include multiple PFTP gateways as well, independent of the HPSS S3 instances. One might imagine a solution where multiple HPSS S3 instances are configured, each with their own PFTP daemon service configured. It could also be determined that there should be multiple HPSS S3 instances per PFTP daemon service, and they could run on the same system.

One consideration for load balancing is that a client must be directed to the same server for its requests in order to support multipart puts (since the server must know about the part information to process the pieces). This can be accomplished using IP source balancing or cookie-based balancing.

IP source balancing may be a good choice if the end clients are expected to come in from a number of source IPs. Otherwise, cookie-based balancing will be needed.

A simple haproxy backend setup using source balancing across four servers might look like:

```
backend app
    balance     source
    server  hpss1 192.168.222.223:8081 check
    server  hpss2 192.168.222.224:8081 check
    server  hpss3 192.168.222.225:8081 check
    server  hpss4 192.168.222.226:8081 check
```

The end user will connect to the HA proxy IP, which will balance requests across the backend servers.

11.14. Enabling HTTPS for S3 proxy

In order to support HTTPS communication, S3 proxy requires generation of an X509 certificate that is placed in a Java key store. The following procedure describes a manual process for creating the Java key store.

1. Determine the interface name to be used for your S3 proxy HTTPS service.

   ```
   Example: s3proxy.mydomain.com
   ```

2. Use the *openssl* utility to create a signing request for your new certificate.

   ```
   $ openssl req -newkey rsa:2048 -sha256 -nodes -out servercert.csr -outform PEM \
         -keyout privkey.pem -subj "\CN=s3proxy.mydomain.com"
   ```

   This example creates a 2048 bit RSA private key and a signing request that can be used to generate a x509 certificate with a SHA-2 256 bit message digest. The private key and message digest types used can be controlled by supplying different command options. See *man req* for more information about this command. The signing request is stored in the *servercert.csr* file in the local directory. The private key is stored in the *privkey.pem* file in the local directory.

3. Send the signing request file to your CA administrator and request that a certificate be created.
4. Bundle the private key and the certificate returned from the CA administrator in to a PKCS12 archive file.

   ```
   $ openssl pkcs12 -export -in servercert.pem -inkey privkey.pem -passout \
          'pass:<storepass>' -name s3proxy > <pkcsfile>
   ```

   For this example, *servercert.pem* is assumed to be the certificate file returned from the CA administrator. *privkey.pem* is assumed to be the private key created in step 2 above. <*storepass*> is the password you select for your key store file. The PKCS12 archive is stored in the file named <*pkcsfile*>.

5. Create a key store using the PKCS12 bundle.

```
$ keytool -importkeystore -srckeystore <pkcsfile> -destkeystore <keystore> \
    -srcstoretype pkcs12 -srcstorepass <storepass> -noprompt \
    -deststoretype pkcs12 -deststorepass <storepass> -alias s3proxy'
```

For this example, <*pkcsfile*> is the file containing the PCKS12 bundle from the previous step. <*storepass*> is the key store password that was selected in the previous step. <*keystore*> is the path name for the new key store to create. The key store is typically created in the */var/hpss/etc* directory and is named something like: *hpss_s3proxy.jks*.

6. Update the S3 proxy configuration to enable SSL and use the keytab file just created. Your configuration file (*/var/hpss/etc/hpss_s3proxy.conf*) should look something like this:

```
#s3proxy.endpoint=http://s3proxy.mydomain.com
s3proxy.secure-endpoint=https://s3proxy.mydomain.com
s3proxy.keystore-path=/var/hpss/etc/hpss_s3proxy.jks
s3proxy.keystore-password=_<storepass>_
```



Notice that the *s3proxy.endpoint* variable is commented out. It is currently recommended to only run either a secure (https) or insecure (http) S3 interface. Running multiple interfaces has not been validated and is likely to cause unexpected results.

7. Restart the S3 proxy.

11.15. Setting up for limited lifetime session tokens

The S3 API supports a capability that allows users to authenticate using a limited lifetime security token obtained using a Keycloak compliant OIDC provider.

11.15.1. Configuring the OIDC provider

To utilize authentication via limited lifetime session token, sites must provide and configure their own OpenID Connect (OIDC) provider. This section describes the required provider configuration.



HPSS provides a sample **configure_keycloak** script that is documented in [configuring_keycloak]. Depending on your requirements, this may be an option for helping with initial OIDC provider configuration. You can also consult the "Configuring Keycloak for production" guide at keycloak.org for help with configuring an OIDC provider.

Consult your chosen OIDC provider's documentation for production configuration guidance. For Keycloak specifically, see the "Configuring Keycloak for production" guide at keycloak.org.

The OIDC provider will require the following:

- Services available via HTTPS communication
- A realm (Keycloak) or other equivalent mechanism configured with all users requiring S3 API access
  - Default realm name used by the S3 API server is *'hpss_s3_realm'*



Realms are Keycloak mechanisms used to partition resources, and may not be supported by other OIDC providers.

- All users requiring S3 API access must have their *'preferred_username'* set to match their HPSS username. If the another claim is used to identify the HPSS user, then that must be indicated by setting the *'s3proxy.authtkn.hpss-username-claim'* configuration value to the claim name.
- A client in the realm (or other mechanism) should be configured for S3 API server use with the following settings:
  - The default client name used by the S3 API server is *'s3proxy-client'*. Other names can be used, but that must be reflected in the S3 API server configuration.
  - Valid redirect URLs for the client set to *'https://<hostname>:<port>/*'*, where <*hostname*> and <*port*> match the values used for the *'s3proxy.authtkn.endpoint'* S3 API server configuration setting.
  - Valid web origins set to the value of *'s3proxy.authtkn.endpoint'* S3 API server configuration setting.
  - Client authentication flow should be set to *'Standard'*



The authentication flow labeled *'Standard'* by Keycloak, refers to the OIDC *'Authorization Code Flow'*. To get the full description you can consult the Keycloak documentation, or expand the help window next to *'Standard flow'* on the client configuration window.

  - Client should support the *'openid'* and *'profile'* scopes
  - Client authorization should be set to *ON* (i.e., private client access enabled)
  - Client secret should match the *'s3proxy.authtkn.client-secret'* configuration value.
  - Client front channel logout should be disabled and client back channel logout should be enabled.
  - Client back channel logout channel should be enabled.
  - Client back channel logout URL should be set to *'https:<hostname>:<port>/logout'*, where <*hostname*> and <*port*> match the values used for the *'s3proxy.authtkn.endpoint'* S3 API server configuration setting.
  - Client should support a RSA RS256 token signing key and X509 certificate
  - Client PKCE should be disabled.

11.15.2. Configuring the S3 API server

Once the Keycloak (or other) OIDC provider is configured, set the appropriate S3 API server

configuration values to match.

s3proxy.authtkn.endpoint

This is the URL supported by the S3 API server token service. It should be set to the same value used for the client redirect URL.



The endpoint URL path should never end with a / character.

s3proxy.authtkn.keystore

This is the key store containing the X509 used for SSL communicate with the S3 API server token service. You can use the same key store as used by the S3 API server (*s3proxy.keystore-path*), or create one for the token service.

s3proxy.authtkn.keystore-password

This is the key store password, set it based on the key store used.

s3proxy.authtkn.oidc-provider-url

This is the URL used by the end user to obtain the initial OIDC ID token. It should set based on your OIDC provider configuration.

s3proxy.authtkn.realm-name

This is the Keycloak realm name associated with the realm to be used by the S3 API server token service. The default value for this configuration parameter is *'hpss_s3_realm'*.

s3proxy.authtkn.client-name

This is the OIDC provider client name used by the end user when obtaining an OIDC ID token. The name is also used by the S3 API server when validating the ID token. This should be set based on the client configuration of your OIDC provider. The default value for this configuration parameter is *'s3proxy-client'*.

s3proxy.authtkn.client-secret

This is the OIDC provider client secret used when making provider requests. This should be set based on the client configuration of your OIDC provider.

s3proxy.authtkn.encryption-key

This is the key used to encrypt session token information maintained on disk storage by the S3 API server token service. A randomly generated AES 256 bit key value should be used for this configuration parameter. Set this parameter to a string with 32 random characters. **Example:** c5D959uuiW6n77jfbSTj2jKtdoKb87JF

s3proxy.authtkn.remote-signing-key

This is the key used to sign requests between made between the S3 API server and the S3 API server token service. A randomly generate HmacSHA256 bit signing key value should be used for this configuration parameter. Set this parameter to a string with 32 random characters. **Example:** opUcAEkmG5XXAx6PFgMLC23jX74bPGZx

s3proxy.authtkn.token-lifetime-hours

This is the number of hours an authentication session will remain valid. The default for this

configuration parameter is 720 hours.

11.15.3. Enabling support for the session token GSS mechanism

The session token GSS mechanism must be enabled on the PFTP daemon system. This is done by copying the mechanism configuration template into the */etc/gss/mech.d* directory. The template can be found in the *config/templates/system_files/mech.hpss.s3tkn.conf* file located in the HPSS installation tree. After copying the file to the *mech.d* directory, edit the file to make sure the specified library path matches the path used for your HPSS installation.

An additional configuration value must be added to the HPSS site configuration on your core machine. The configuration variable is named **"S3 Access Token Authenticator URL"**. The value of this variable allows the GSS mechanism to validate session tokens. The URL used should match what is configured for the *s3proxy.authtkn.endpoint* parameter.

The following example shows an HPSS site configuration for an S3 API server token service running at port 8000 of the *myhost.mysite.com* server.

```
$ cat /var/hpss/etc/site.conf
HPSS Site Configuration = {
    Site Name = myhost
    UNIX Server Certificate = /var/hpss/etc/login_service_cert.pem
    UNIX Server Key = /var/hpss/etc/login_service_cert.key
    UNIX CA File = /var/hpss/etc/login_service_cert.pem
    UNIX Server Address = myhost.mysite.com
    UNIX Server Port = 1024
    S3 Access Token Authenticator URL = https://myhost.mysite.com:8000
    Realms = {
        MYSITE.COM = {
            Directory = unix
            PAM Stack = hpss
            Auth Type = unix
            Realm ID = 10000
            Passwd = /var/hpss/etc/passwd
            Group = /var/hpss/etc/group
            Shadow = /var/hpss/etc/shadow
        }
    }
}
```

11.15.4. Configuring the HPSS PFTP daemon

An additional configuration parameter is required for the HPSS PFTP Daemon section of the HPSS.conf file to enable clear channel communications for the use of S3 API session tokens. The following example shows a *HPSS.conf* that has clean channel communications enabled.

```
$ cat /var/hpss/etc/HPSS.conf
PFTP Daemon = {
```

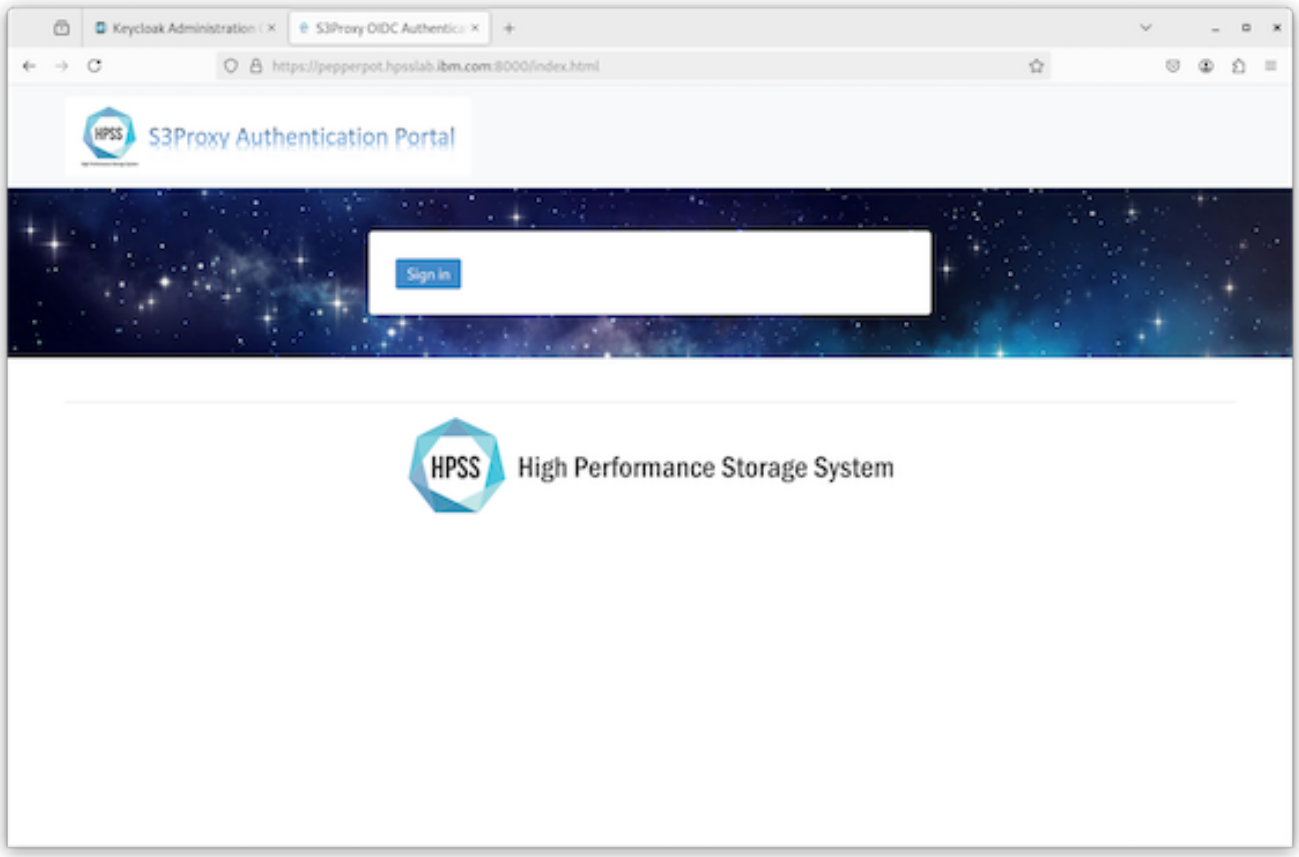
```
        # Allow the Daemon to take Core Dumps
        Allow Core Files
...

        # Allow the CCC Command to the GSS Daemon
        # This allows the client to request a cleartext control channel
        Allow CCC Command
...
```

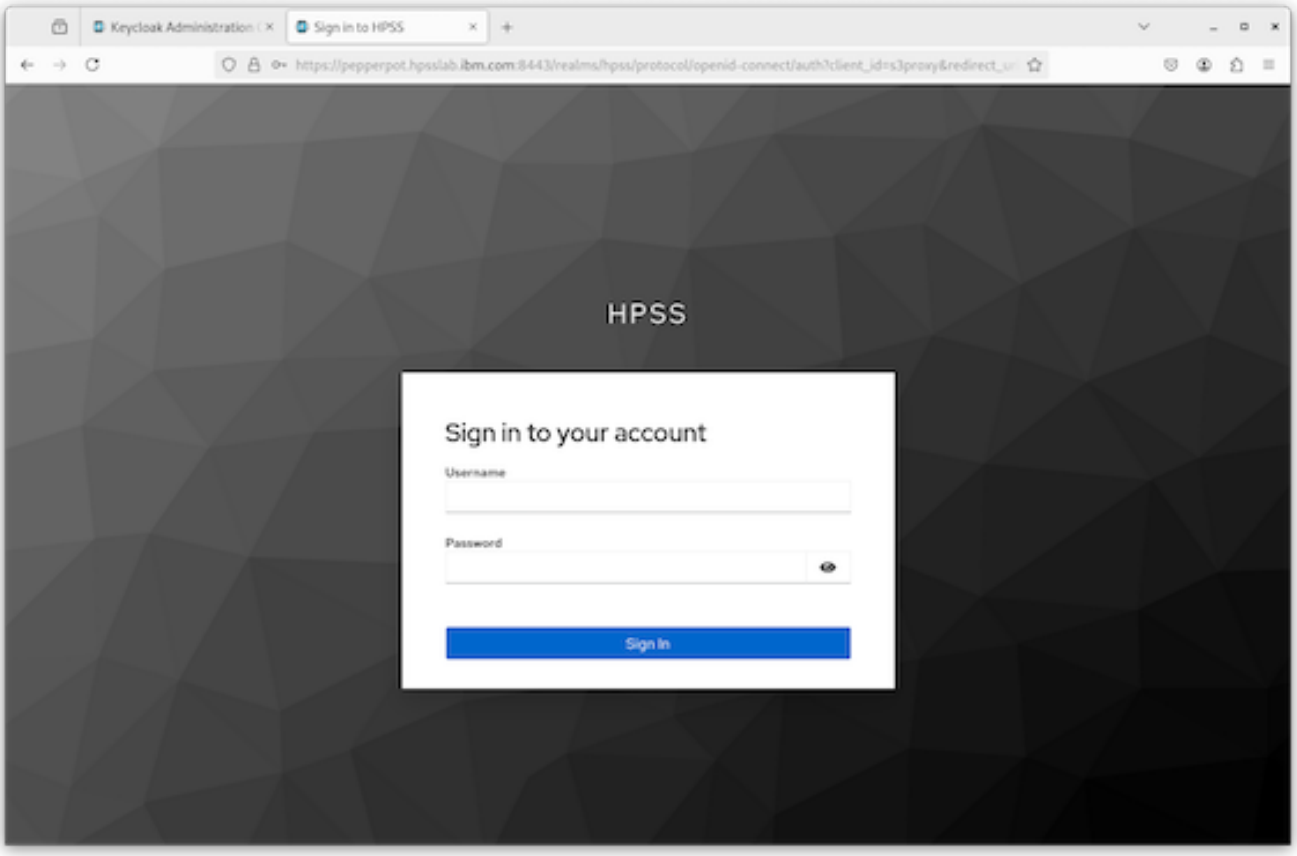
It is possible to enable the S3 API session token mechanism to use privacy protected channel communications. This is accomplished by setting the *hpsspftp.s3tkn.privacy* configuration variable to *true*. However, be aware that this has the possibility of negatively effecting S3 API performance.

11.15.5. Obtaining a session token

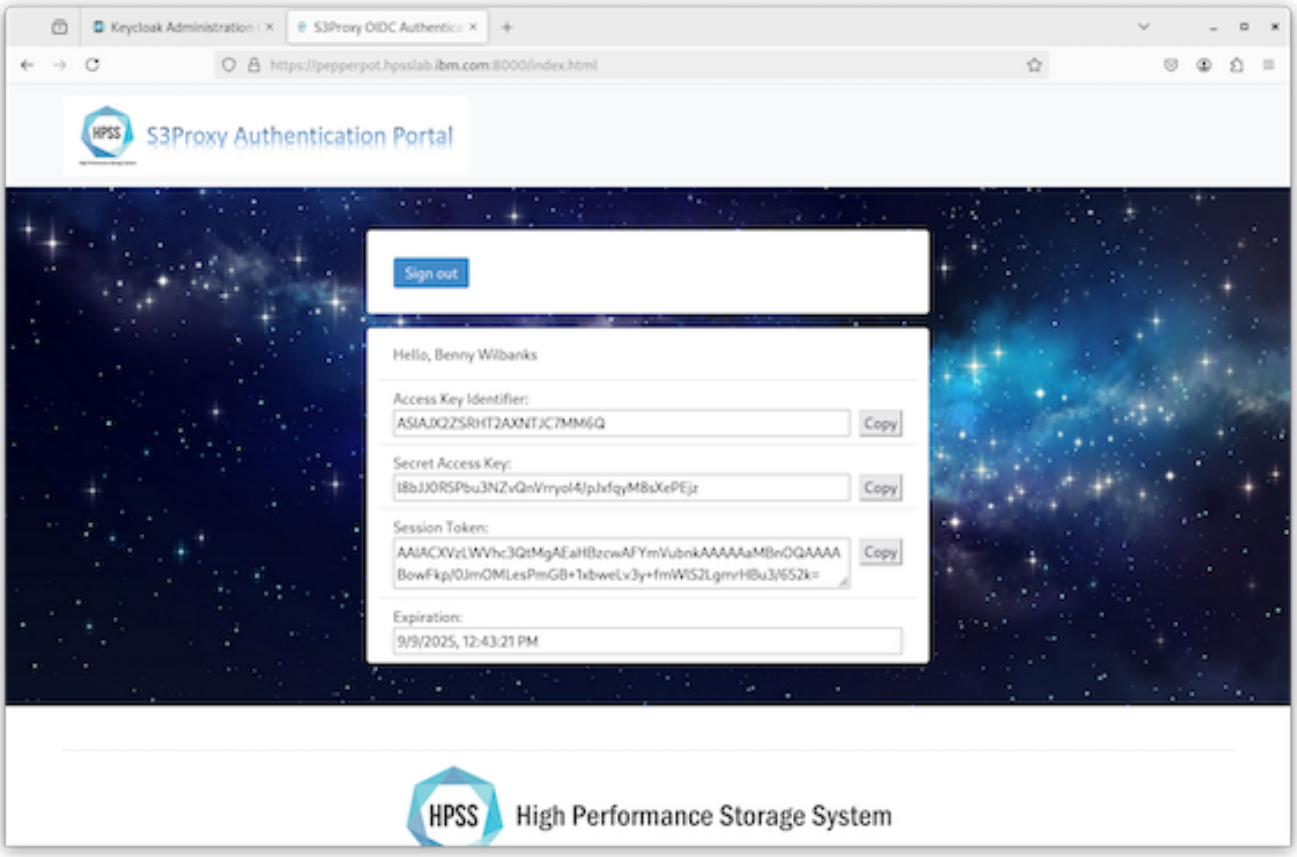
To obtain a session token the user points a browser at the configured S3 API server session token endpoint (https:/myhost.mydomain.com:8000 for this example):



The user requests a login via the *Sign in* buttone, and is redirected to authorization server (i.e., Keycloak or other OIDC provider) and is prompted to provide their user name and password.



Once the user is authenticated, the AWS style authentication information is displayed. The user can then copy the fields to the system clipboard and paste them to an S3 client interactive login prompt, or to a S3 client configuration file.



11.15.6. Login helper

The login helper is a mechanism that can be used to integrate with the "AWS SDK Process Credential provider". This capability provides a way to extend the credential provider chain for HPSS S3 API use cases. The login helper can be configured as a credential provider by referencing it from the AWS Config file *credential_process* setting. For example, the following can be used to configure the S3 API login helper as the credential provides for AWS SDK requests that use the *hpss-s3* profile.

```
$ cat ~/.aws/config
...
[profile hpss-s3]
credential_process = s3proxy_login_helper -d https://<hostname>:<port>
...
```

The login helper will launch a web browser, allowing the user to obtain and OIDC ID token from the configured providers. The OIDC ID token is then used to obtain a limited life-time S3 API session token. The helper will cache the session token under the users home directory (~.aws/s3proxy_credentials), and provide it when requested by the AWS SDK.

The login helper wheel can be obtained from the hpss/config directory and installed using the Python pip tools.

```
sudo -E python -m pip install /opt/hpss/config/s3proxy_login_helper-12.1-py3-none-
any.whl
```

11.15.6.1. Headless Mode

For situations where the client machines have only headless access and a web browser cannot be accessed, the login helper can be configured to run in a "headless" mode. In this cases, instead of starting a web browser when a new session token is needed, the helper will prompt the user via the current pseudo terminal with a URL. The user can cut-n-paste the URL into a web browser running on another machine and complete login processing.

For headless mode, the login helper requires a x509 certificate. The certificate can be created via the following steps.

1. Create a headless certificate.

   ```
   $ openssl req -newkey rsa:2048 -sha256 -nodes -out helpercert.csr -outform PEM
   ```

2. Have the nice CA admin sign your cert.

   ```
   $ scp helpercert.csr niceca@cahost:/cadir/X509CA/helpercert.csr
   ```

3. Get the original request and the signed cert back.

```
$ scp niceca@cahost:/cadir/X509CA/helpercert.tar .
$ tar xvf helpercert.tar
./helpercert.pem
./helpercert.csr
```

4. Combine the signed cert and the private key.

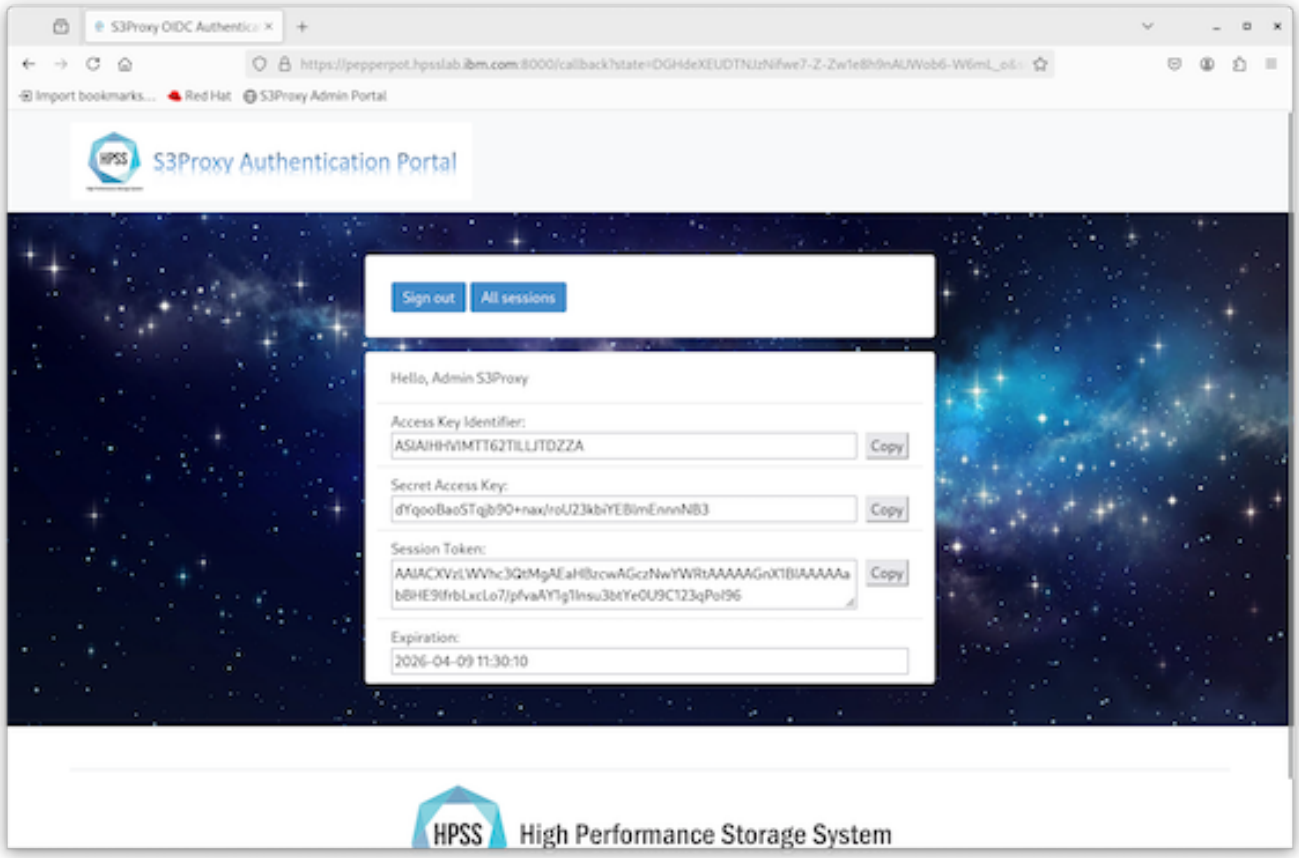
```
cat ./helpercert.pem ./privkey.pem > ./helpercert.crt
```

5. Use the -H option with to run the helper in "headless" mode.

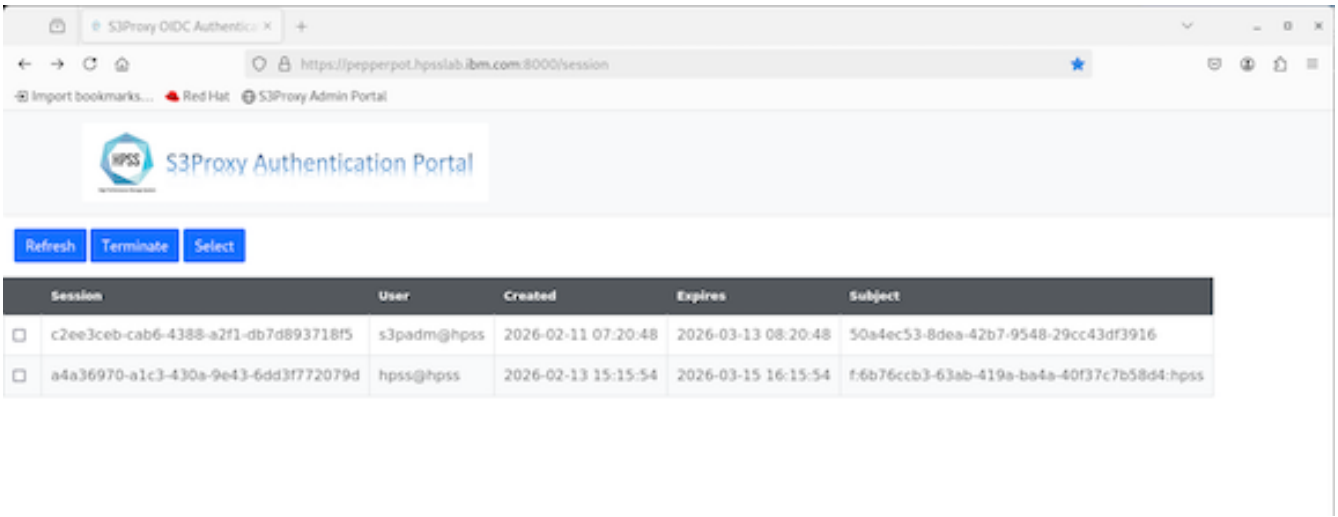
```
$ cat ~/.aws/config
...
[profile hpss-s3]
credential_process = s3proxy_login_helper -d -H ./helpercert.crt
https://<hostname>:8666
...
```

11.15.7. Managing sessions

The S3 Proxy session tokens are independent from the OIDC sessions, and if configuration dictates, can out live the OIDC session expiration. While the corresponding OIDC session exists, S3 Proxy sessions and associated tokens can be revoked by signing out of the session from the OIDC administrative console. However, if the OIDC session has expired any current S3 Proxy session can be revoked using the S3 Proxy session management console. The session management console can be accessed by logging in as the S3 Proxy administrative user and selecting **"All sessions"**.



This will take you to a window listing all the active S3 Proxy sessions.



This interface allows the S3 Proxy administrator, or operator, to list all of the active S3 Proxy sessions. Administrative users can then select and terminate any desired sessions.

11.15.7.1. Session management configuration

Session management requires configuration of one or more users with the S3 Proxy administrative (and if needed, the operator) roles. Users with the S3 Proxy administrator role can list and terminate sessions, while users with the S3 Proxy operator role can only list sessions.

The concept of roles is not a standard across all OIDC providers, so the following configuration settings allow for roles to be supported via claim extensions.

s3proxy.authtkn.session-adm-role-claim

This is the name of the claim in the ID token that is used to identify any user role or group association. This claim should be a JSON array of role or group membership for the user.



- Keycloak roles can be configured using the **"Realm Roles→Create Role"** window.
- User role assignment can be done via the **"Role mapping"** tab of the **"Users→User detail"** window.
- The roles can be included in the ID token by going to the **"Client scopes"** window and typing "roles" into the search bar. Select the **"roles"** client scope and the **"Mappers"** tab. From this tab you can add a builtin mapper of type **"User Realm Role"**, then open its configuration and make sure the **"Add to ID token"** switch is turned on.

s3proxy.authtkn.session-adm-role-adm

This is the role or group name associated with an S3 Proxy administrator. This name identifier must be present in the role claim JSON array for the user to have administrative privilege.

s3proxy.authtkn.session-adm-role-opr

This is the role of group name associated with an S3 Proxy operator. This name identifier must be present in the role claim JSON array for the user to have operator privilege.

11.15.8. Integrating Keycloak with existing systems

For existing HPSS systems, maintaining large numbers of users both in the HPSS directory (*passwd* file for Unix authentication) and in your OIDC provider could be seen as cumbersome. If you currently use Unix authentication for your HPSS system, and wish to give your users S3 API access, then ask your support representative about the HPSS storage provider for Keycloak. This provider allows Keycloak to access your HPSS *passwd* and *group* files. Your support representative can provide more information on obtain and configuring the provider.

11.16. Setting up for Kerberos use

The S3 API is specified to use signatures based on shared secrets to authenticate documents. Intended for sites which do not allow shared secrets (passwords) to be stored in databases or files, the S3 proxy adds another option: Kerberos. This allows use of one's credential without saving plaintext passwords into a database. However, use of this feature has some caveats.

Since AWS does not specify Kerberos as an authentication method, there are no compatible off-the-shelf AWS client programs that can use Kerberos authentication. The only current way to create requests is to use more generic web tools such as cURL, possibly as a standalone tool with scripting, or as an embedded library in one's program. Examples of such use exist in the `src/s3proxy/kerberos_examples` subdirectory.

To set up Kerberos, first a machine service name is required. The service name is in the form `HTTP/machine.host.name@REALM`. The host name used in the principal must be the fully qualified, canonical machine name, as this is what the client will be attempting to use. An example run for

the Kerberos admin for S3 on system `s3.example.com` might look like:

```
$ kadmin
kadmin:  add_principal +allow_svr -randkey HTTP/s3.example.com@EXAMPLE.COM
...
kadmin:  ktadd -k s3_http.keytab HTTP/s3.example.com@EXAMPLE.COM
```

This will place the keytab for `HTTP/s3.example.com` in the file `s3_http.keytab`. You will need to copy or move that file into a place where the S3 proxy can read it, and set the S3 proxy's keytab path in `hpss_s3proxy.conf` to point to that location. You will also need to configure the Kerberos realm correctly. Finally, you will need to set the authorization type to Kerberos.

A snippet of the `hpss_s3proxy.conf` file might look like:

```
s3proxy.authorization=KERBEROS
s3proxy.kerberos.realm=EXAMPLE.COM
s3proxy.kerberos.ktname=/var/hpss/etc/s3_http.keytab
```

Finally, it is essential that the credential be delegatable. By default, Kerberos is not configured to ask for forwardable credentials. One must explicitly ask for them with `kinit -f <my.cred>` or by modifying `/etc/krb5.conf`. The libdefaults section in `krb5.conf` would need to exist and must have at least this setting:

```
[libdefaults]
 forwardable = true
```

Now, when running cURL, the following command line options will be required for each invocation:

```
 --negotiate              Uses the GSS/Negotiate mechanism, which becomes Kerberos
 --delegation always      Tells GSS to delegate your credentials to the server
 -u :                     cURL will not perform authentication without a user name
                          and password, even though GSS ignores both. Therefore a
                          colon can be used, indicating empty username and empty
                          password.
```

Chapter 12. HPSS configuration overview

12.1. Introduction

This chapter defines the high-level steps necessary to configure, start, and verify correct operation of a new HPSS system, whether that system is created from scratch or created by conversion from any previous version of HPSS.

To create or modify the HPSS configuration, we recommend that the administrator first be familiar with the information described in the HPSS basics and HPSS planning sections.

Before performing the procedures described in this chapter, be certain that the appropriate system preparation steps have been performed. See System preparation section for more information. For a system created from scratch, be certain that the HPSS installation and infrastructure configurations have been completed. See the HPSS installation and infrastructure configuration section for more information. To convert from a previous system, see the conversion steps provided to you by HPSS support.

12.2. Starting the SSM GUI for the first time

The HPSS system is ready to be configured using SSM once the HPSS software is installed on the node and the HPSS infrastructure components are configured.

Before starting SSM for the first time, ensure that:

1. Logging has been configured using `/opt/hpss/config/configure_logging` (see Configure Logging in the Installation chapter)
2. The `/var/hpss/log` directory exists and is writable by the HPSS user
3. The rsyslog service has been restarted after configuration



If logging is not configured, SSM will still start, but you will need to check `/var/log/messages` *for HPSS log entries instead of* `/var/hpss/log/HPSS.log`*. Alternatively, you can configure SSM to look for logs in a different location by modifying the* `ssm.conf` *file.*

In order to start the SSM GUI you must first start all infrastructure components and the SSM System Manager as follows:

```
% /opt/hpss/bin/rc.hpss -m start
```

Next you will need to add an SSM Admin user. To do this you will need to invoke the hpssuser(7) utility as follows:

```
% /opt/hpss/bin/hpssuser -add hpss -<unix|krb> -ssm
```



The above commands must be done as root.

Once the SSM Admin user has been created, you can invoke the SSM GUI as follows (for `hpssgui.pl` options, see the hpssgui(7) man page):

```
% /opt/hpss/bin/hpssgui.pl
```

Note: *This command may be done as an HPSS user.*

When the SSM GUI is running you can begin to configure the rest of HPSS (servers, devices, and so on) as described in the following sections. For more information on SSM, refer to Using SSM.

12.3. HPSS configuration roadmap (new HPSS sites)

The following steps summarize the configuration of an HPSS system from scratch (not upgrading from a previous release). It is important that the steps be performed in the order listed. Each step is required unless otherwise indicated. Each step is discussed in more detail in the referenced section.

1. Configure storage subsystems (see Creating a new storage subsystem)

   *Subsystems can be configured only partially at this time. The **Gatekeeper**, **Default COS**, and **Allowed COS** fields will be updated in a later step.*

2. Configure HPSS storage policies
   - Accounting Policy (refer to Accounting Policy window)
   - Log Policies (refer to Log policies)
   - Migration Policies (refer to Migration policies)
   - Purge Policies (refer to Purge policies)
3. Configure HPSS storage characteristics
   - Storage Classes (refer to Configured Storage Classes window)
   - Storage Hierarchies (refer to Storage hierarchies)
   - Classes of Service (refer to Classes of Service)
4. Configure HPSS servers (refer to Server configuration)
5. Create global configuration (refer to Global Configuration window)
6. Configure MVR devices and PVL drives (refer to Configure a new device and drive)
7. Configure file families, if used (refer to File families)
8. Update storage subsystem configurations with Gatekeeper and COS information (refer to

Modifying a storage subsystem and Storage Subsystem Configuration window)

9. Create the endpoint map (refer to Core Server additional configuration)

12.4. Initial HPSS startup roadmap (all sites)

This section provides instructions for starting the HPSS servers and performing post-startup configuration. For sites which are converting from a previous version of HPSS, only Step 1 may be necessary. For sites configuring a new system from scratch, all steps are necessary:

1. Start the HPSS servers (refer to Starting HPSS servers)
2. Unlock the PVL drives (refer to Locking a drive)
3. Create HPSS storage space:
   a. Import volumes into HPSS (refer to Importing volumes into HPSS)
   b. Create storage resources (refer to Creating storage resources)
4. Create additional HPSS users (refer to Adding HPSS users)
5. Create Filesets and Junctions (refer to Filesets & Junctions List and Creating a junction)
6. Create HPSS `/log` directory

   If log archiving is enabled, using an HPSS name space tool such as hpssmkdir(7) or **ftp**, create the `/log` directory in HPSS. This directory must be owned by *hpsslog* and have permissions `rwxr-xr-x`.

   The `/log` directory can be created by the root user using **ftp** as follows:

```
% ftp <node> <HPSS Port> # login as root user
ftp> mkdir /log
ftp> quote site chown hpsslog /log
ftp> quote site chmod 755 /log
```

   If Account Validation is enabled, principal "hpsslog" must appear in the ACCTVAL table with a default account code allowing write access to the `/log` directory. The utility hpss_avaledit(7) can be used to verify and, if necessary, correct this configuration.

```
% hpss_avaledit -list
ACCOUNT ACCOUNT_NAME DEFAULT USER
...
301 hpsslog yes hpsslog
...
```

   If "hpsslog" does not appear in the list, or if the DEFAULT value is "no", or the user name is something other than "hpsslog", do the following:

```
% hpss_avaledit -remove hpsslog -all # if necessary
```

```
% hpss_avaledit -add -uname hpsslog -aname hpsslog -def
```

12.5. Additional configuration roadmap (all sites)

This section provides a high level roadmap for additional HPSS configuration.

1. Configure HPSS user interfaces (refer to User interfaces).
2. Set up backup for DB2 and other infrastructure (refer to Backup and recovery).
3. Set up High Availability, if desired (see the HPSS planning section).
4. Optionally configure support for both authentication mechanisms (refer to the Supporting both UNIX and Kerberos authentication for SSM section).

12.6. Verification checklists (all sites)

This section provides a number of checklists regarding configuration, operational, and performance issues.

12.6.1. Configuration checklists

After HPSS is running, the administrator should use the following checklists to verify that HPSS was configured or converted correctly.

Global configuration

- Verify that a **Default Class of Service** has been selected.
- Verify that a **Root Core Server** has been selected.

Storage subsystem configuration

- Verify that a **Default Class of Service** has been selected if desired.
- Verify that a **Gatekeeper** has been selected if gatekeeping or account validation is required.
- Verify that the **COS Name** list has been filled in correctly.
- Verify that a **Core Server** and **Migration/Purge Server** have been configured for each storage subsystem.
- Verify that each storage subsystem is accessible by using lsjunctions(7) and ensuring that there is at least one junction to the Root fileset of each subsystem. (The root fileset for a given subsystem can be found in the specific configuration for the subsystem's Core Server.)

Servers

- Verify that all required HPSS servers are configured and running.
- Verify that all servers are configured with the intended level of logging, whether using their server-specific policies or the default policy. Also verify that all Core Server and Mover log policies have the DEBUG flag turned on to aid in the diagnostics of any future problems.

Devices and drives

- Verify that all devices/drives are configured and each is assigned to an appropriate PVR or Mover.
- For tape devices, verify that the **Locate Support** option is enabled if supported.
- For tape devices, verify that the **NO-DELAY** option is enabled if supported by the device.
- For disk devices, verify that the **Capacity** and **Starting Offset** values are correct.
- Verify that all configured drives are unlocked.

Storage classes

- Verify that all storage classes are defined and each has sufficient free storage space.
- Verify that each storage class that will be migrated and purged is configured with the appropriate migration and purge policy.
- Verify that no storage class at the lowest level in a hierarchy is configured with a migration or purge policy.
- To support repack and recover of tape volumes, verify that the stripe width of each tape storage class is less than half of the number of available drives of the appropriate drive type.

Storage hierarchies

- Verify that all storage hierarchies are defined.

Classes of Service

- Verify that all Classes of Service are defined.
- Verify that each Class of Service (COS) is associated with the appropriate storage hierarchy.
- Verify that the COS is configured to use the characteristics of the hierarchy and the underlying storage classes. In addition, verify that the Classes of Service have the correct **Minimum File Size** and **Maximum File Size** values. If these sizes overlap, the file placement may be indeterminate when the user creates a file using the size hints. For Classes of Services which are not to be used as part of standard file placement, set their **Force Selection** flag to ON so that they will only be chosen if specified by their **COS ID**.
- Verify that Classes of Service with multiple copies have the **Retry Stage Failures from Secondary Copy** flag enabled.

File families, filesets, and junctions

- Verify that file families and filesets are created according to the site's requirements.
- Verify that each fileset is associated with the appropriate file family, COS, or both.
- Verify that each fileset has an associated junction.

User interfaces

- Verify that the desired HPSS user interfaces (such as FTP, PFTP, Client API, or any others) are configured (refer to User interfaces)

12.6.2. Operational checklists

The administrator should follow these checklists to ensure that HPSS is operating properly.

Configured user interfaces

- Create files of various sizes on each defined COS.
- Verify that the files are created in the expected storage class with acceptable transfer rates.
- If necessary, redefine the associated storage class definition to enhance the throughput performance.
- Update the characteristics fields (such as **Transfer Rate**, **Latency**, and others) in the storage class and Class of Service definition to reflect actual performance results.
- After the files are created on the correct storage class, verify that the files are created with correct file ownerships and permissions.
- Verify that other file operations (such as **delete**, **chmod**, and **copy**) work.
- If accounting is configured, verify that the files are created with the correct accounting indices.
- If file families, filesets, and junctions are configured, verify that they work as intended.

Devices and drives

- Ensure that each drive mounts, reads, and writes.

Storage management

- Verify that migration and purge operations work as intended.
- Start Migration and Purge operations manually to verify that files are migrated and purged correctly.
- Verify that files can be accessed after being migrated/purged.
- Monitor free space from the top level storage class in each hierarchy to verify that the migration and purge policy are maintaining adequate free space.

12.6.3. Performance checklist

Measure data transfer rates in each COS for:

- Client writes to disk
- Migration from disk to tape
- Staging from tape to disk
- Client reads from disk

Transfer rates should be close to the speed of the underlying hardware. The actual hardware speeds can be obtained from their specifications and by testing directly from the operating system (for example, by using the Linux **dd** command to read and write to each device). Keep in mind that transfer performance can be limited by factors external to HPSS. For example, HPSS file read performance may be limited by the performance of the UNIX file system writing the file rather than limits inside HPSS.

Chapter 13. Security and system access

13.1. Security services

The approach to security within HPSS divides services into two APIs, known as mechanisms, each of which has multiple implementations. Configuration files control which implementation of each mechanism is used in the security realm (analogous to a DCE cell) for an HPSS system. Security mechanisms are implemented in shared object libraries and are described to HPSS by a configuration file. HPSS programs that need to use the mechanism dynamically link the library to the program when the program starts.

The first type of mechanism is the **authentication mechanism**. This API is used to acquire credentials and to verify the credentials of clients. Authentication verifies that a client really is who it claims to be.

The second type of mechanism is the **authorization mechanism**. Once a client's identity has been verified, this API is used to obtain the UID, GID, group membership, and other authorization details associated with the client that are used to determine the privileges accorded to the client and the resources to which it has access.

13.1.1. Security services configuration

Ordinarily, the configuration files that control HPSS's access to security services are set up either by the installation tool **mkhpss** or by the metadata conversion tools. This section is provided purely for reference.

- `/var/hpss`

  When installing HPSS on a new system, **mkhpss** will create the **hpss** user with the home directory as `/var/hpss`.

Each of the files below is stored by default in `/var/hpss/etc`.

- `authz.conf`

  This file defines which shared libraries provide implementations of the authorization mechanisms. It is a plain text file. Each line is either a comment beginning with # or consists of two fields separated by whitespace: the path to a shared library and the name of the function used to initialize the security interface.

- `site.conf`

  This file defines security realm options. This is a plain text file in which each line is a comment beginning with # or defines a combination of authentication and authorization for a realm, as well as encryption parameters. See the man page for `site.conf` for details.

- `mech.hpss.conf`

This file defines the GSS configuration for HPSS UNIX security. This file should be installed into the GSS mech directory (usually `/etc/gss/mech.d/`) on the system where the HPSS UNIX authentication service will run. The file defines the location of the HPSS unix authentication library. The template for this file is in `config/templates/system_files/mech.hpss.conf`.

- PAM `hpss` configuration file

  This file will be automatically created by **mkhpss** or during conversion. This file defines the PAM configuration for HPSS UNIX security. This file should be installed in the PAM directory (usually `/etc/pam.d/`) on the system where the HPSS UNIX authentication service will run. Sites can configure this file based on their desired PAM settings.

13.1.2. Security mechanisms

HPSS supports UNIX and Kerberos mechanisms for authentication. It supports LDAP and UNIX mechanisms for authorization.

13.1.2.1. UNIX

UNIX-based mechanisms are provided both for authentication and authorization. These can draw either from the actual UNIX user and group information on the current host or from a separately maintained set of files used only by HPSS. This behavior is controlled by the setting of the stanza "Use System UNIX" in the `/var/hpss/etc/site.conf` file. If this stanza appears, the system's UNIX user and group data will be used. Otherwise, local files created and maintained by the HPSS UNIX utilities will be used. Consult the man pages for each utility for details of its use.

Like with Kerberos, there is a service, the HPSS UNIX Login Service (**hpss_unix_service**), which is used for authentication functions. The UNIX utilities now depend on this service for their operations, no longer depending on setuid for such permissions.

The HPSS UNIX Login Service can authenticate credentials and also provide temporary credentials for user-to-service connections. To reduce the need for connections to the logon service, the tickets are cached in the user's kernel keyring on the local system. The tickets can be listed by the **hpss_unix_ticket** command. For the moment, there is no dedicated HPSS tool for deleting them. The workaround is to use the **keyctl** system command. See the **hpss_unix_ticket** man page for information about both commands.

Keep in mind that the user and group databases must be kept synchronized across all nodes in an HPSS system. If using the actual UNIX information, this can be accomplished using a service such as NIS. If using the HPSS local files, these must manually be kept in synchronization across HPSS nodes.

HPSS UNIX utilities

- hpss_unix_keytab(7) is used to define "keytab" files that can be used to acquire credentials recognized by the UNIX authentication mechanism.
- hpss_unix_user(7) is used to manage users in the HPSS password file (`/var/hpss/etc/passwd`).
- hpss_unix_group(7) is used to manage users in the HPSS groups file (`/var/hpss/etc/group`).
- hpss_unix_passwd(7) is used to change passwords of users in the HPSS password file.

- hpss_unix_auth(7) is used to test the authentication system. The command-line options allow for testing authentication scenarios at various levels.
- [hpss_unix_ticket] is used to print the UNIX tickets that are known to the system UNIX user.

HPSS UNIX Login Service

- The **hpss_unix_service** is a systemd service (hpss_login.service)
- **hpss_unix_service** is an HPSS prerequistite service. It is automatically started when running **rc.hpss** on the core server.

13.1.2.2. UNIX and NIS or LDAP

When setting up HPSS on a system with network-based accounting systems such as NIS (Network Information Service) or LDAP (Lightweight Directory Access Protocol), it is still necessary to keep the HPSS server accounts local to the system. This is because the current keytab system for UNIX uses a hashing method to convert the account's password into a keytab. To test whether a keytab works, HPSS must be able to get the account's password hash from the passwd database and hash it to compare with the keytab hash. LDAP typically depends on authenticating to the LDAP server instead of conducting the comparison; hence, it does not even provide the hashes.

In addition to authentication difficulties, this also means that there are two locations for account information to be obtained. HPSS at present does not have a standard way to search "HPSS" local password files for some accounts while using LDAP for users. Because of this, the recommended approach for integrating NIS or LDAP into HPSS is the following:

- use UNIX authentication, and configured to use system files
- store HPSS server accounts and any users that will use keytabs (that is, with hpssadm(7)) in the local system
- if using the HPSS accounts across multiple systems, copy the HPSS password and shadow entries and keytabs manually from the core system to its ancillary systems in order to maintain the same hashes.

13.1.2.3. Kerberos 5

Kerberos 5 is an option for the authentication mechanism. When this option is used, the local realm name is taken to be the name of a Kerberos realm. The Kerberos security services are used to obtain and verify credentials.

13.1.2.4. LDAP



LDAP authorization is not supported by IBM Service Agreements. The following information is provided for sites planning to use LDAP authorization with HPSS as a site-supported feature.

An option for the authorization mechanism is to store HPSS security information in an LDAP directory. LDAP is a standard for providing directory services over a TCP/IP network. A server supporting the LDAP protocol provides a hierarchical view of a centralized repository of data and provides clients with sophisticated search options. The LDAP software supported by the HPSS LDAP

authorization mechanism is OpenLDAP. One advantage of using the LDAP mechanism over the UNIX mechanism is that LDAP provides a central repository of information that is used by all HPSS nodes; it doesn't have to be manually kept in sync.

The rest of this section deals with how to accomplish various administrative tasks if the LDAP authorization mechanism is used.

Working with principals

Creating a principal

A principal is an entity with credentials, like a user or a server. The most straightforward way to create a new principal is to use the **-add** and **-ldap** options of the hpssuser(7) utility. The utility will prompt for any needed information and will drive the **hpss_ldap_admin** utility to create a new principal entry in the LDAP server. To create a new principal directly with the **hpss_ldap_admin** utility, use the following command at the prompt:

```
princ create -uid <uid> -name <name> -gid <gid> -home <home> -shell
<shell> [-uuid <uuid>]
```

If no UUID is supplied, one will be generated.

Deleting a principal

Likewise, use the **-del** and **-ldap** options of the hpssuser(7) utility to delete the named principal from the LDAP server. To delete a named principal directly with the **hpss_ldap_admin** utility, use the following command at the prompt:

```
princ delete [-uid <uid>] [-name <name>] [-gid <gid>]
[-uuid <uuid>]
```

You may supply any of the arguments listed. This command will delete any principal entries in the LDAP information that have the indicated attributes.

Working with groups

Creating a group

To create a new group, use the following command at the **hpss_ldap_admin** prompt:

```
group create -gid <gid> -name <name> [-uuid <uuid>]
```

If no UUID is supplied, one will be generated.

Deleting a group

To delete a group, use the following command at the **hpss_ldap_admin** prompt:

```
group delete [-gid <gid>] [-name <name>] [-uuid <uuid>]
```

You may supply any of the arguments listed. This command will delete any group entries in the LDAP information that have the indicated attributes.

Adding a member to a group

To add a principal to a group, use the following command at the **hpss_ldap_admin** prompt:

```
group add <principal> [-gid <gid>] [-name <name>] [-uuid <uuid>]
```

You may supply any of the arguments listed to select the group to which the named principal will be added.

Removing a member from a group

To remove a principal from a group, use the following command at the **hpss_ldap_admin** prompt:

```
group remove <principal> [-gid <gid>] [-name <name>] [-uuid <uuid>]
```

You may supply any of the arguments listed to select the group from which the named principal will be removed.

Working with trusted foreign realms

Creating a trusted foreign realm

To add an entry for a trusted foreign realm, use the following **hpss_ldap_admin** command:

```
trealm create -id <realmID> -mech <mechanism> -name <realmName> -url
<url>
```

The arguments are as follows:

- *-id <realmID>* is the numeric realm ID for the foreign realm.
- *-mech <mechanism>* is a string identifying the authorization mechanism in use at the foreign realm, such as "unix" or "ldap".
- *-name <realmName>* is the name of the foreign realm; for example, "SOMEREALM.SOMEDOMAIN.COM".
- *-url <url>* is the URL of the security mechanism of the foreign realm. This only matters if the foreign realm is using LDAP as its authorization mechanism. If so, this must be the LDAP URL of the main entry for the security realm in the foreign LDAP server. This should be obtained from the foreign site's administrator. An example would be: `"ldap://theirldapserver.foreign.com/cn=FOREIGNREALM.FOREIGN.COM"`.

Deleting a trusted foreign realm

To delete an entry for a trusted foreign realm, use the following **hpss_ldap_admin** command:

```
trealm delete [-id <realmID>] [-name <realmName>]
```

Any of the arguments listed can be supplied to select the trusted realm entry that will be deleted.

13.2. HPSS server security ACLs

Beginning with release 6.2, HPSS uses a table of access control information stored in the DB2 configuration database to control access to HPSS servers. This is the AUTHZACL table. HPSS software uses the configured authentication mechanism (for example, Kerberos) to determine a caller's identity via credentials provided by the caller, then uses the configured authorization mechanism to retrieve the details of the caller that determine the access granted. Once the identity and authorization information have been obtained, each HPSS server grants or denies the caller's request based on the access control list information stored in the database.

The default ACLs (Access Control Lists) for each type of server are as follows:

Core Server:

```
r--c--- user ${HPSS_PRINCIPAL_FTPD}
rw-c-dt user ${HPSS_PRINCIPAL_MPS}
rw-c-d- user ${HPSS_PRINCIPAL_SSM}
r--c--- user ${HPSS_PRINCIPAL_FS}
------t any_other
```

Gatekeeper:

```
rw----- user ${HPSS_PRINCIPAL_CORE}
rw-c--- user ${HPSS_PRINCIPAL_SSM}
------t any_other
```

Mover:

```
r--c--t user ${HPSS_PRINCIPAL_PVL}
rw-c--t user ${HPSS_PRINCIPAL_SSM}
r-----t any_other
```

PVL:

```
rw-c-dt user ${HPSS_PRINCIPAL_CORE}
rw--dt user ${HPSS_PRINCIPAL_PVR}
rw-c-dt user ${HPSS_PRINCIPAL_SSM}
------t any_other
```

PVR:

```
rw-c-dt user ${HPSS_PRINCIPAL_PVL}
rw-c--t user ${HPSS_PRINCIPAL_SSM}
```

```
------t any_other
```

RAIT Engine:

```
r--c--t user ${HPSS_PRINCIPAL_PVL}
rw-c--t user ${HPSS_PRINCIPAL_SSM}
r-----t any_other
```

SSM:

```
rwxcidt user ${HPSS_PRINCIPAL_ADM_USER}
------t any_other
```

All other types:

```
rw-c-dt user ${HPSS_PRINCIPAL_SSM}
------t any_other
```

In most cases, the ACLs created by default for new servers should be adequate. In normal operation, the only ACL that has to be altered is the one for the SSM client interface. This is handled automatically by the **-ssm** option of the hpssuser(7) utility. If, for some reason, an ACL should need to be modified in some other way, the hpss_server_acl(7) utility can be used. See the hpss_server_acl(7) man page for more information.

13.3. SSM user security

SSM supports two types of users, administrators and operators:

administrator

This security level is normally assigned to an HPSS administrator. Administrators may open all SSM windows and perform all control functions provided by SSM.

operator

This security level is normally assigned to an HPSS operator. Operators may open most SSM windows and can perform all SSM control functions except for HPSS configuration.

Security is applied both at the window level and the field level. A user must have permission to open a window to do anything with it. If the user does succeed in opening a window, all items on that window may be viewed. Field level security then determines whether the user can modify fields, push buttons, or otherwise modify the window.

The security level of an SSM user is determined by his entry in the access control information table in the HPSS configuration database. The initial security level for a user is assigned when the SSM user is created by hpssuser(7). Security levels may be viewed and modified with the hpss_server_acl(7) utility. See also SSM user authorization.

13.4. Restricting user access to HPSS

13.4.1. Overview

System administrators may deny access to various HPSS operations for a specific user by including that user in the Restrict Access File. This file is read by the Core Server at server startup. The file can also be reread by the Core Server by selecting **Refresh Restricted Access Table** from the **Operations** menu on the SSM GUI.

13.4.2. Configuration File

To set up the configuration file, add the `HPSS_RESTRICT_ACCESS_FILE` environment variable to `/var/hpss/etc/env.conf`. Set the value of this variable to the path name of the file that will contain the list of access restrictions. Ensure this environment variable is defined appropriately before starting the Core Server. To notify the Core Server of updates to the value of this environment variable, reinit the Core Server.

For example:

```
HPSS_RESTRICT_ACCESS_FILE=/var/hpss/etc/restrict-access-file
```



The file should be configured on the system where the root Core Server is running. Additionally, if running with multiple storage subsystems on different machines, be sure to configure the `HPSS_RESTRICTED_ACCESS_FILE` on each machine where a Core Server runs.

13.4.2.1. File Permissions

Examine the file system permissions for the Restrict Access File. You'll want to set the permissions to be permissive enough for the Core Server to read the file. And you'll want to set the permissions to be restrictive enough so that the file cannot be modified by people without authority to (un)restrict end-users' access to HPSS.

13.4.2.2. File Format

The Restrict Access File format is:

- one entry per user
- one user per line
- each line uses a CSV format that lists a user ID and one or more restrictions
- each line can end with a comment beginning with the `#` character

In general the file format looks like:

```
<UID> <restriction_1>, <restriction_2>, ···, <restriction_n> #comment
```



Certain types of errors in a Restrict Access File can cause the Core Server to exit during startup. These errors are typically misspelled restriction names or lines that are syntactically incorrect. In addition, finding a user ID in more than one line (more than one config file entry) will cause the Core Server to exit.

Please test any changes to the Restrict Access File prior to making the update on a production HPSS system. If an update to the Restrict Access File causes the Core Server to exit, revert to a previous version of the Restrict Access File and examine the updates for syntax errors.

13.4.3. ra_verify_config

The tool ra_verify_config is provided to, hopefully, detect and eliminate the types of errors mentioned above before these errors are discovered by the Core Server. ra_verify_config reads each line of the Restrict Access File and attempts to verify that the line is correct. It is expected that this verification will be done before submitting the file to the Core Server.

13.4.4. Refresh Restricted Access Table

Selecting **Refresh Restricted Access Table** under the **Operations** menu on the SSM GUI causes the Core Server to reread the Restrict Access File in order to refresh the restricted access table. No messages will be displayed for success or failure. If successful, an event message will be generated for each user in the Restrict Access File indicating the user ID and the restrictions for that user. If unsuccessful, one or more alarms will be generated indicating the cause of the failure.

13.4.5. Restriction Types

Individual restrictions are listed for each user in the restrict access config file. The restriction names and meanings are:

RestrictAll	restrict the user from all actions
RestrictFileCopies	restrict the user from copying files
RestrictFileCreates	restrict the user from creating new files
RestrictFileWrites	restrict the user from writing to new and existing files
RestrictStages	restrict the user from staging files from lower levels (typically tape) to higher levels (typically disk) in storage hierarchies

13.4.6. Example Restrict Access File

```
4200 RestrictAll         # don't allow user ID 4200 to perform any HPSS operations
4042 RestrictFileCreates # don't allow user ID 4042 to create any new files (write
access to existing files is OK, though)
4242 RestrictStages      # don't allow user ID 4242 to stage any data from tape
4420 RestrictFileCopies, RestrictFileWrites, RestrictFileCreates # don't allow user ID
```

```
4420 to create any new data in HPSS
```

Chapter 14. Using SSM

14.1. The SSM System Manager

14.1.1. Starting the SSM System Manager

Before starting the SSM System Manager (SM), review the SM key environment variables described in the Storage System Management section.

If the default values are not desired, override them using the hpss_set_env(7) utility.

To start the SM, invoke the rc.hpss(7) script as follows:

```
% su -
% /opt/hpss/bin/rc.hpss -m start
```

14.1.2. Tuning the System Manager RPC thread pool and request queue sizes

Tuning the System Manager RPC thread pool and request queue sizes can improve the performance of both the System Manager and its clients. It is not necessary, however, to do the tuning when bringing up SSM for the first time. In fact, it can be helpful to postpone the tuning until after the site has a chance to learn its own SSM usage patterns.

The System Manager client interface RPC thread pool size is defined in the **Thread Pool Size** field on the Interface Controls tab of the System Manager's *Core Server Configuration* window. This is the maximum number of RPCs that can be active at any one time for the client interface (that is, all SSM clients). For the server RPC interface (connections to the SSM System Manager from other HPSS servers), this value is determined by the HPSS_SM_SRV_TPOOL_SIZE environment variable.

The System Manager client interface RPC request queue size is defined in the **Request Queue Size** field on the Interface Controls tab of the System Manager's *Core Server Configuration* window (see Interface controls).

This is the maximum number of RPC requests from clients which can be queued and waiting to become active. For the server RPC interface, this value is determined by the HPSS_SM_SRV_QUEUE_SIZE environment variable.

Ideally, if the site runs many clients, the client interface RPC thread pool size should be as large as possible; the default is 100. Testing this value at 300 showed the System Manager memory size more than doubled. The larger RPC thread pool size makes the System Manager much more responsive to the many clients but also requires it to use more memory.

Experimentation shows that leaving the client interface RPC thread pool size at 100 and leaving the client interface RPC request queue size at its default (600) works pretty well for up to about 40 clients. During further experimentation, setting the client interface RPC request queue size to 1000 resulted in very little effect on memory usage; with 40 clients connected, the client interface RPC

request queue used never went above about 500, but the client interface RPC thread pool was constantly filled.

Avoid allowing the client interface RPC thread pool to become full. When this happens, new RPCs will be put into the client interface RPC request queue to wait for a free thread from the thread pool. This makes the client response appear slow, because each RPC request is having to wait its turn in the queue. To help mitigate this, when the thread pool is full, the System Manager notifies all the threads in the thread pool that are waiting on list updates to return to the client as if they just timed out as normal. This could be as many as 15 threads per client that are awakened and told to return, which makes those threads free to do other work.

If the client interface RPC thread pool is still full (as it could be if, for example, there were 15 threads in the client interface RPC request queue that took over the 15 that were just released), then the System Manager sets the wait time for the new RPCs to one second rather than whatever the client requested. This way the RPC won't try to hang around too long.

Realize that once the System Manager gets in this mode (constantly having a full client interface RPC thread pool and having to cut short the thread wait times), the System Manager starts working hard and the CPU usage will start to increase. If you close some windows, some clients, or some of both, things should start to stabilize again.

You can see whether the System Manager client interface RPC thread pool has ever been full by looking at the **Maximum Active/Queued RPCs** field in the Client column of the RPC Interface Information group in the *System Manager Statistics* window (see System Manager Statistics window).

If this number is greater than or equal to the corresponding client interface's Thread Pool Size (default 100), then the thread pool was full at some time during the System Manager execution (although it may not be full currently).

To tell whether the thread pool is currently full, look at the number of Queued RPCs. If Queued RPCs is zero, then the thread pool is not full at the moment.

If Active RPCs is equal to Thread Pool Size, then the thread pool for the interface is currently full. Active RPCs should never be greater than Thread Pool Size. When it reaches the Thread Pool Size, then the new RPCs will be queued and Queued RPCs become greater than zero.

When the thread pool gets full, the System Manager tries harder to clear them out before accepting new ones, to ensure that if the thread pool fills up, it doesn't stay full for long.

If the site runs with low refresh rates and more than 40 clients, the recommendation is to set the client interface RPC thread pool size to 150 or 200 and the client interface RPC request queue size to 1000 in the System Manager *Server Configuration* window (see Interface controls).

Otherwise, the default values should work well.

14.1.3. Labeling the System Manager RPC program number

Labeling the System Manager RPC program number is not required but can be a useful debugging aid.

The SSM System Manager registers with the RPC portmapper at initialization. As part of this registration, it tells the portmapper its RPC program number. Each HPSS server configuration contains the server's RPC program number. To find the System Manager's program number, open the *Servers* window, select the SSM System Manager, and click the **Configure** button to open the *SSM System Manager Configuration* window. The System Manager's RPC program number is in the **Program Number** field on the Execution Controls tab of this window.

The **rpcinfo** utility with the **-p** option will list all registered programs, their RPC program numbers, and the port on which they are currently listening for RPCs. When diagnosing SSM problems, it can be useful to run the **rpcinfo** program and search for the System Manager RPC program number in the output. Look to see whether the System Manager has successfully initialized its RPC interface and to see which port clients must access to reach the System Manager. Note that the WebSSM communicates using the SSMAPI, so it does not need direct access to the SSM server.

This task can be made a bit easier if the System Manager RPC program number is labeled in the portmapper. To do this, add a line for the System Manager in the `/etc/rpc` file specifying the program number and a convenient RPC service name such as "hpss_ssm" (note that names may not contain embedded spaces). Then this service name will show up in the **rpcinfo** output.

The format of the `/etc/rpc` file differs slightly across platforms. See the platform-specific man pages for the `rpc` file for details. The **rpcinfo** utility is typically found in `/usr/sbin`.

14.2. Configuration and startup of the SSM API and Web SSM UI

HPSS offers a REST (Representational State Transfer) API for the SSM, which utilizes HTTP requests to access and manipulate data. This section describes in detail the configuration and startup of the SSM API. It also explains the proper setup of SSM users with your OIDC provider, and how to interact with the SSM API and its endpoints. This section also includes instructions on how to configure the Web SSM UI, which provides the means to display SSM data in a web browser.

14.2.1. Overview

SSM API

This is the set of HPSS Python modules that uses FastAPI endpoints to provide HPSS SSM services. It defines the REST API endpoints and communicates directly to the SSM server, translating requests between HPSS structures and JSON.

Web SSM UI

An HPSS Python module that runs a FastAPI web application that displays SSM statistics provided by the SSM API.

FastAPI

A high performance web framework for building APIs with Python. This framework facilitates the creation of REST endpoints.

nginx

Web server software that serves the SSM API endpoints.

Uvicorn

A Python Asynchronous Server Gateway Interface (ASGI) web server implementation.

Gunicorn

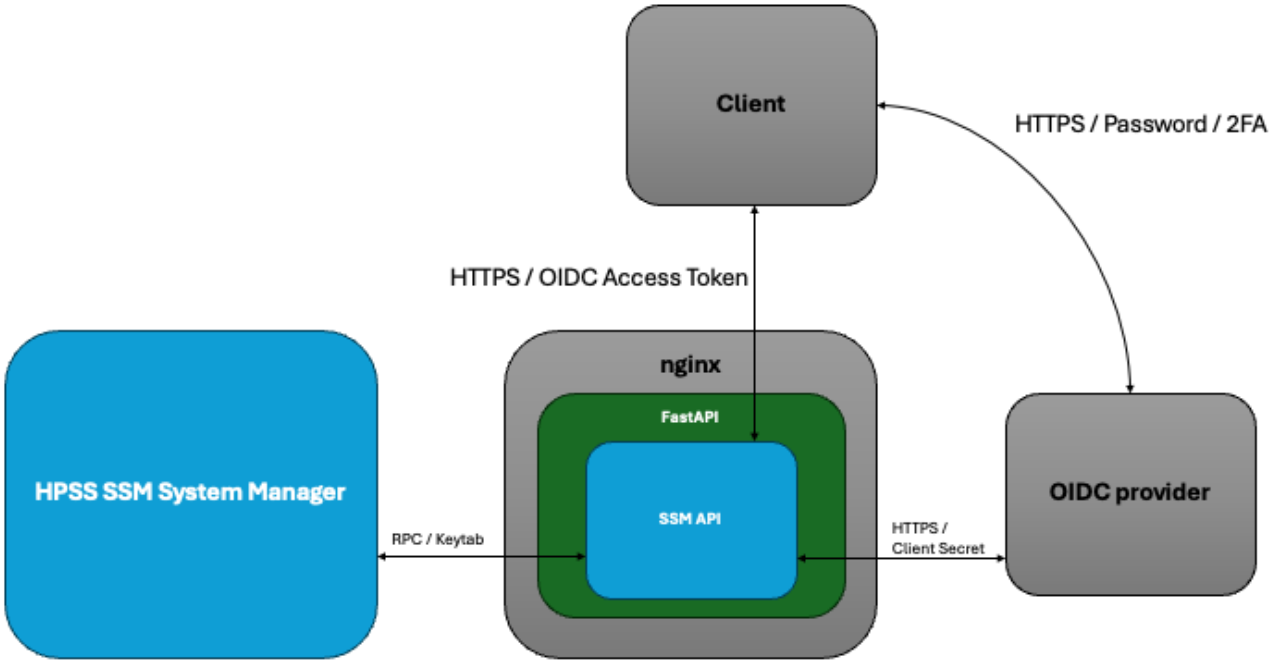
Green Unicorn, which essentially serves as a process manager for managing Uvicorn worker processes.

OpenID Connect (OIDC)

An authentication protocol built on top of the OAuth 2.0 framework that verifies user identities for access to protected endpoints.

OIDC Provider

Software that provides OIDC services. Sites must provide their own OIDC provider and configure it with the SSM API.



This diagram describes the layout of the SSM API. The SSM API is a solution that runs as a FastAPI application within nginx. The SSM API application communicates directly with the SSM server, translating between HPSS data structures and JSON/HTTP.

The OIDC provider provides authentication services for SSM API. The client will log into the OIDC provider with their credentials. Then, the client will send a token to the SSM API as part of the request. The SSM API will use the OIDC to check that the token is valid and what role is available to the user (admin, operator).

14.2.2. Configuring nginx for Reverse Proxying

To configure nginx, you will need to run the `$HPSS_ROOT/config/configure_nginx` script. This script automates the creation of a virtual host for nginx, which will proxy requests to the SSM API and the Web SSM UI. Additional configuration will be required based on the site's OIDC provider.

Note, before running the script nginx should be installed on the system as discussed in the Prerequisite software overview section.

Once ready, run the script as root or with elevated system privileges. The script will prompt for necessary information such as the server name (the domain part of a URL), the port, the OIDC server, OIDC port, and SSL key and certificate file locations. This part of the guide assumes you have installed the SSL certificates and have their locations in hand as discussed in the Install SSL certificates for web services section.

Example

```
/opt/hpss/config/configure_nginx \
   -s example.com \
   -k /etc/ssl/private/example.pem \
   -c /etc/ssl/certs/example.pem \
   -f \
   -d example.com \
   -w
```

Note, the -w configure WebSSM option configures nginx to reverse proxy for that service. The configure_webssm script still needs to be run to configure WebSSM.

Note, the default ports are port 80 for unencrypted traffic and port 443 for encrypted traffic. The nginx service setup on the Core Server should be used only for HPSS services; no other HTTP services (such as Apache) should be running on this server.

After the script completes, it is good practice to test the nginx configuration before restarting to ensure no errors exist. This can be done by running:

```
% /usr/sbin/nginx -t
```

14.2.3. Quick startup of the SSM API

To quickly set up and configure the SSM API, run the `$HPSS_ROOT/config/configure_ssmapi` script to create the necessary system services along with corresponding configurations, and configure the SSM API for the OIDC provider.

The `$HPSS_ROOT/config/configure_ssmapi` script will create a environment file /var/hpss/ssm/hpss_ssmapi.env.

Example

```
/opt/hpss/config/configure_ssmapi \
   -c f4BNpAGIfdo6KaXS67MKhpTVCdt26DDW \
   -a 90 \
   -k
```

The /var/hpss/ssm/hpss_ssmapi.env file contains critical OIDC configuration parameters that must be properly configured for your site's OIDC provider:

OIDC_CLIENT_ID

The client ID assigned by your OIDC provider for the HPSS SSM API application.

OIDC_CLIENT_SECRET

The client secret assigned by your OIDC provider. This must match the secret configured in your OIDC provider for the HPSS SSM API client/application.

OIDC_SERVER_METADATA_URL

The OpenID Connect discovery endpoint URL for your OIDC provider. This is typically in the format https://your-oidc-provider/.well-known/openid-configuration.

OIDC_USER_CLAIM

The claim field in the OIDC token that contains the username. Common values are "preferred_username", "email", or "sub". This must match the username format used in HPSS SSM user accounts.

Additionally, you must configure the callback URL in your OIDC provider to allow redirects back to the SSM API after authentication. The callback URL should be: https://your-server/ssmapi/oauth/callback



The callback URL must be configured for authentication to work. If it is missing or does not match the registered redirect URI, all authentication attempts will fail with a redirect_uri error because the identity provider cannot redirect the user back.

Also note the DB_NAME, DB_SCHEMA, and DB_KEYTAB parameters of the /var/hpss/ssm/hpss_ssmapi.env file. These should match the site's configuration database, and can be adjusted to match the site's configuration.

Consult your OIDC provider's documentation for specific instructions on obtaining these values and configuring the client application.

After the $HPSS_ROOT/config/configure_ssmapi script completes and all OIDC parameters are set correctly, the hpss_ssmapi service should be installed and ready to start.

You can start and check its status with:

```
# systemctl start hpss_ssmapi
# systemctl status hpss_ssmapi
```

FastAPI automatically generates interactive API documentation including request/response schemas, validation rules, and example data. The FastAPI documentation is accessible through a web browser at the SSM API Docs Endpoint.

14.2.4. SSM API Configuration Details

The `hpss_ssmapi.conf.py` file is a configuration script for Gunicorn. This file is set up using the `$HPSS_ROOT/config/configure_ssmapi` script. It will automatically configure an optimal worker count using a formula to take advantage of multi-core CPUs to run multiple processes in parallel. The formula is half of the available CPU cores rounded down plus one for redundancy purposes.

By default, the Gunicorn process will bind to the server socket 127.0.0.1:8000. If a different server socket is desired, the *bind* variable must be changed in the `hpss_ssmapi.conf.py` file.

It may not be necessary to tune the API with the default configuration, but adjusting the worker count can improve performance and resource efficiency. The default calculation provides a balanced approach, ensuring enough processes to handle incoming requests without overloading the system. It accounts for available CPU resources while maintaining stability, making it a practical choice for most deployments.

However, any changes to the performance profile of the SSM API should be tested rigorously before being deployed. This includes capturing performance metrics and evaluating resource utilization. Additionally, regression testing should be performed to confirm that optimizations or modifications do not introduce unintended side effects or degrade existing functionality with HPSS.

By default, the SSM API will log to files in `/var/hpss/log`. The `configure_logging` script should automatically configure the recommended log rotation policy.

See Recommendations for syslog configuration for details on the SSM API logging configuration.

14.2.5. SSM API Docs Endpoint

FastAPI has a built-in documentation endpoint that constructs documentation for REST API endpoints from the endpoint definitions. This endpoint is the primary means for understanding each individual REST SSM API in terms of:

- What classes of operations are available
- What operations are available
- What the expected inputs, outputs, and error responses are for each endpoint
- Endpoint examples

The location of the `/docs` endpoint will be based on the host configuration for nginx. For example, if your web server URL is https://example.com, then two important SSM API docs endpoints are:

- https://example.com/ssmapi/docs
- https://example.com/ssmapi/latest/docs

The `ssmapi/latest/docs` endpoint documents the latest version of the SSM APIs while the

`ssmapi/docs` endpoint documents:

- OAUTH endpoints for authentication with the OIDC provider
- Links to the `docs` endpoint for each available version of the SSM API
- A metrics endpoint for getting access to prometheus-style metrics on endpoint usage

14.2.5.1. SSM API Docs and Versioning

The SSM API uses *versioned endpoints*. This means that when significant changes are made to the meaning of an endpoint, the SSM API will place that under a new version of that endpoint while retaining the old inputs and output (to the extent possible) in the older version.

Each version has its own `/docs` endpoint describing that particular version. For example, version 1 is documented in `ssmapi/v1/docs`. The latest version is always available at `ssmapi/latest/docs`.

14.2.6. Client interface setup

See the Install Web Service Prerequisites section for information on requirements for the SSM API.

Web-based APIs are designed to expose specific endpoints (also technically called routes) that correspond to certain resources or operations. Clients can send HTTP requests to these endpoints to perform actions like retrieving data, creating new resources, updating existing ones, or deleting them.

A common way to interact with web-based APIs is by using tools like cURL. However, regardless of the tool, the items needed to interact with the API will be:

1. **The API endpoint.** The base URL of the API where the requests are sent (e.g., https://example.com/ssmapi/v1/servers).
2. **Access Token.** A valid access token obtained through the security mechanism. See Creating API Access Tokens.
3. **Authorization Headers.** The authorization headers (or heads) must be included in the HTTP request heads in the `Authorization: Bearer <access_token>` format. The access token can be retrieved from the `/security/me` endpoint when logged in.
4. **HTTP Method.** The type of request being made (e.g., `GET`, `POST`, `DELETE`, etc.)
5. **Request Body (if applicable).** For requests such as `POST` requests, include the data payload in the required JSON format.
6. **Query Parameters (if applicable).** Key-value pairs appended to the URL.

14.2.7. Creating SSM API Users

Because the SSM API uses OIDC for authentication, every SSM user who should have access to the SSM API must be a valid user in the site's OIDC provider. Additionally, every OIDC user who wants to interact with the SSM API must be a valid SSM user with either an operator or admin role, created as described in The **hpssuser** utility.

Sites must ensure that users who need access to SSM are provisioned in the organization's OIDC

identity provider and are granted access to the SSM application. The SSM API relies on this OIDC provider for authentication. Consult your identity provider's documentation for instructions on creating, managing, and assigning users.



The username of the HPSS SSM user must exactly match the username of the user in the configured OIDC identity provider.

14.2.8. Creating API Access Tokens

To generate an access token, start by logging into your account via your OIDC provider, accessible at https://example.com/ssmapi/oauth/login. Upon successful login, navigate to https://example.com/ssmapi/v1/docs in your browser. Scroll down to the *security* section to find the *access_token* endpoint. Clicking on it should reveal various configuration options for the access token, including its duration and privilege level.

This page will create the token and will be displayed on the screen in JSON format. Be sure to copy and securely store it, as it will not be shown again. For security reasons, treat your access tokens like a password - avoid sharing it and store it in a secure location.

14.2.9. Examples: Using the SSM API

Several additional examples can be found in the SSM API docs, which provides extensive examples for each endpoint using cURL.

Scenario:

You want to retrieve information about a class of service from an SSM API endpoint.

Without a SSL verification

```
curl -X GET "https://example.com/ssmapi/v1/classes-of-service/1002" \
     -H "access-token: eyJahbab1aAhjaha81009...." \
     -H "Accept: application/json" -k
```

With a custom certificate authority (CA) store

```
curl -X GET "https://example.com/ssmapi/v1/classes-of-service/1002" \
     -H "access-token: eyJahbab1aAhjaha81009...." \
     -H "Accept: application/json" --cacert <certificate_path>
```

Alternatively the environment variables CURL_CA_BUNDLE could be set to the path of your CA bundle, or you could set SSL_CERT_FILE to the path of your certificate, or SSL_CERT_DIR to the directory containing the certificate.

Explanation:

- `-X GET`: Specifies the HTTP method (GET in this case).

- `"https://example.com/ssmapi/v1/classes-of-service/1002"`: The URL includes a specific endpoint for COS with the path parameter `1002` being the COS ID.
- `-H "access-token: eyJahbab1aAhjaha81009...."`: The access token in the header. This is the token generated from Creating API Access Tokens.
- `--cacert`: Tells curl to use the specified certificate file to verify the peer. The file may contain multiple CA certificates. The certificate(s) must be in PEM format. See the curl man page for more details.

Scenario:

You want to create a new file family from an SSM API endpoint using a Python script.

```
import requests

# Define the URL for the HTTP request
url = "https://example.com/ssmapi/v1/file-families"

# Define the JSON payload
payload = {
   "Id": 1000,
   "Name": "A new file family"
}

# Set the request headers
headers = {
   "Content-Type": "application/json",
   "Authorization": "Bearer eyJahbab1aAhjaha81009...."
}

# Perform the POST request
response = requests.post(url, json=payload, headers=headers)

# Print the response of the POST request
print("Status Code: ", response.status_code)
print("Response Body: ", response.json())
```

Note, the provided Python code uses the **requests** HTTP library for Python.

14.2.10. Managing SSM API Sessions and API Keys

SSM API Sessions

Sessions are limited to 24 hours and initiated by a login to the OIDC provider.

Users can list their active sessions by logging into their OIDC dashboard.

Users can destroy their session token using the logout endpoint (for example https://example.com/ssmapi/oauth/logout). This will destroy the active session token.

Administrators can destroy user sessions if they have sufficient access within OIDC to revoke

user sessions.

SSM API Keys

API keys have a user-defined duration, and can be listed or revoked by the administrator using the ssmapi_token(7) tool. Users currently cannot revoke their own API keys.

14.2.11. Current limitations

The SSM API currently supports many basic monitoring and configuration operations. Several features found in the hpssgui(7) may not yet be available as an SSM API endpoint. More information on what is currently available can be found on the SSM API docs page.

Additionally, the SSM API does not support PATCH operations on endpoints, meaning partial updates of resources are not yet allowed.

14.2.12. Quick startup of the Web SSM UI

See the Install Web Service Prerequisites section for information on requirements for the Web SSM.

To quickly set up and configure the Web SSM UI, run the `$HPSS_ROOT/config/configure_webssm` script, which will install FastAPI and other required Python dependencies, and copy the necessary system service along with corresponding configurations. After its completion, the service should be installed and ready to start. You can start and check its status with:



Web SSM requires the SSM API to be properly configured with the OIDC identity provider for authentication to work.

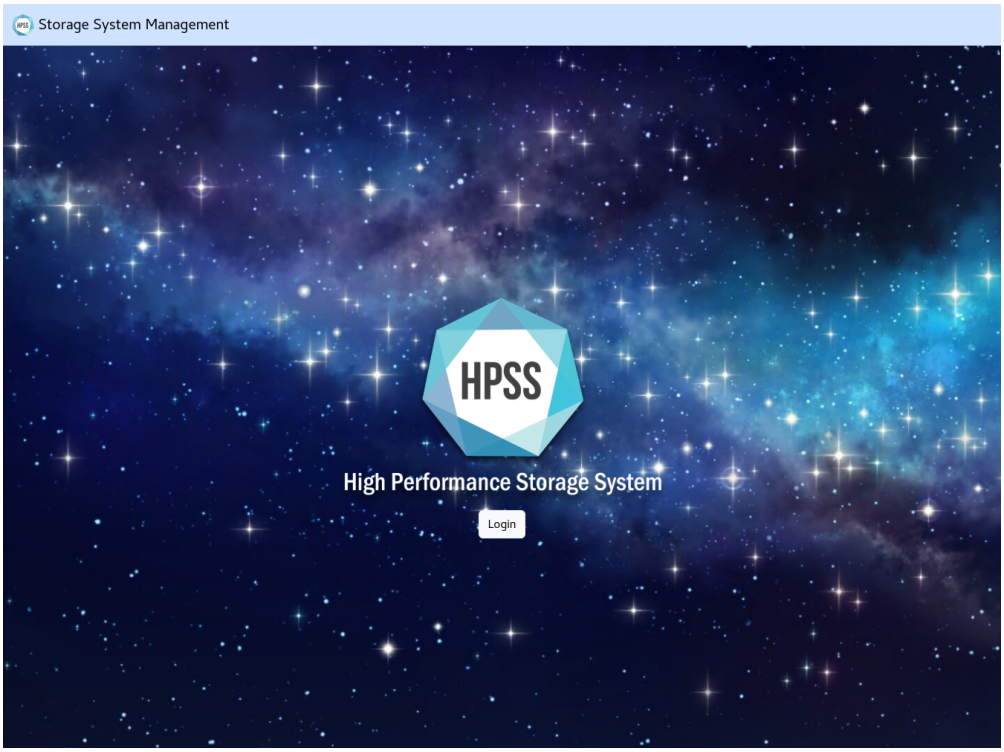
```
# systemctl start hpss_webssm
# systemctl status hpss_webssm
```

The `$HPSS_ROOT/config/configure_webssm` script will create a environment file /var/hpss/ssm/hpss_webssm.env. This file will have a SSMAPI_URL variable set. This variable must match the URL of an SSM API server providing endpoints the Web SSM client can use to access SSM data.

The location of the Web SSM UI will be based on the host configuration for nginx. For example, if your web server URL is https://example.com, then the interface will be at https://example.com/webssm.

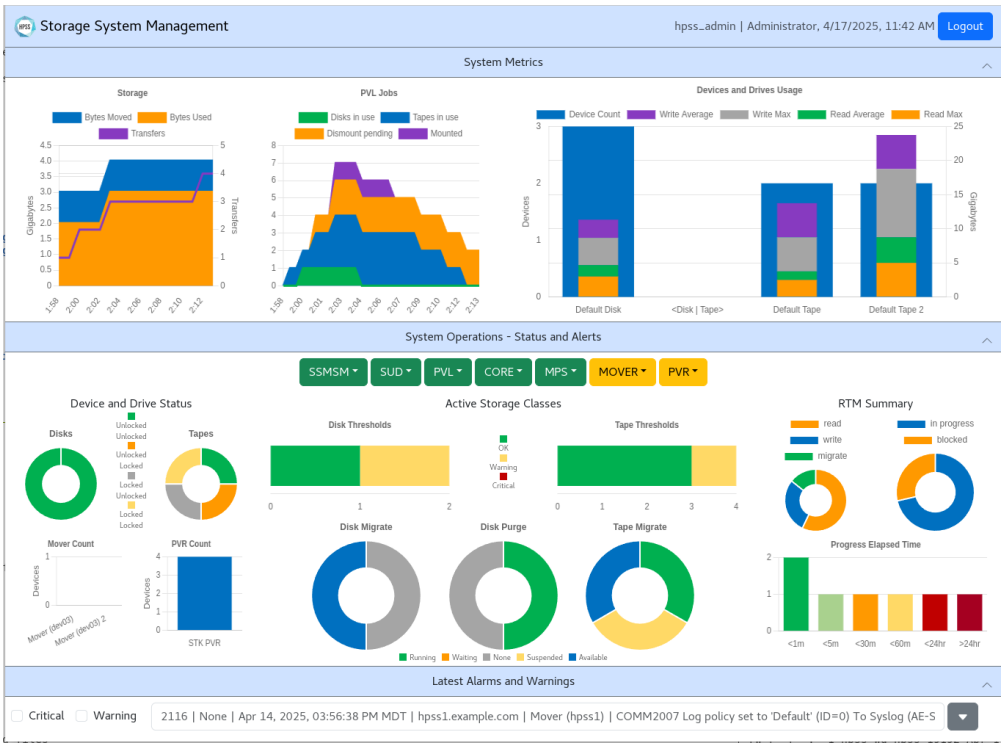
14.2.13. Using the Web SSM UI

When the page initially loads after accessing the Web SSM UI, users will be directed to a login screen.



After clicking on the "Login" button, users will be redirected to the OIDC login screen as configured by the SSM API.

After successfully authenticating, users will be directed to the main Web SSM UI:



The main interface contains several charts in two rows, which are collapsible on click.

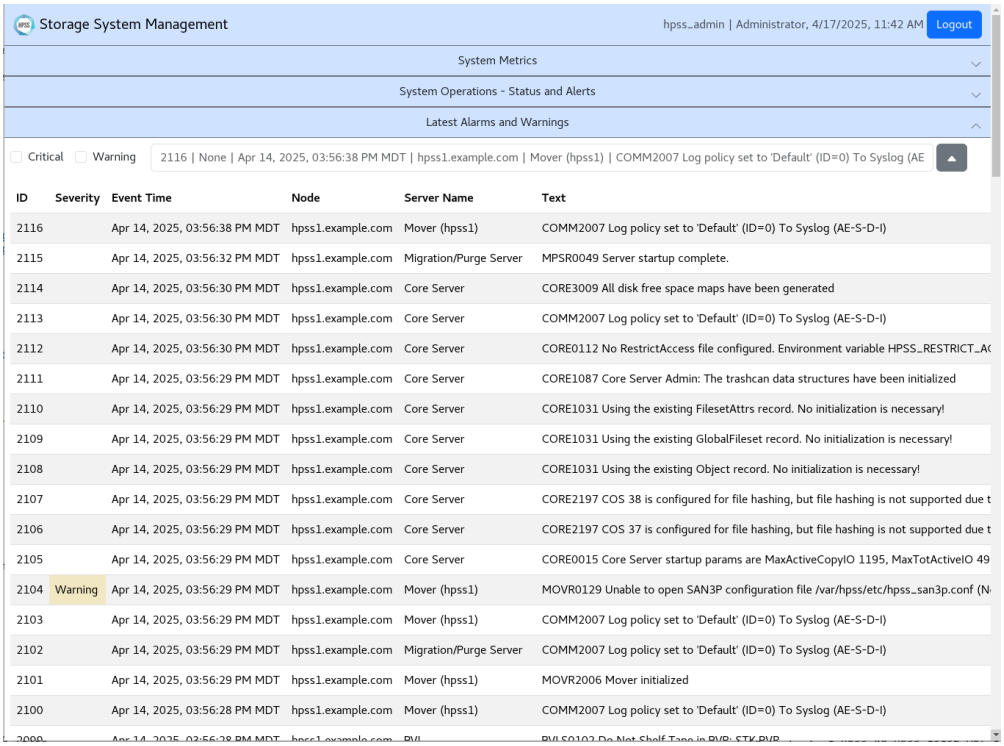
System Metrics:

- Storage: Bytes used, bytes moved, and transfers performed since last SSM restart.
- PVL Jobs: Disks in use, tapes in use, dismounts pending, and mounted volumes.

- Devices and Drives Usage: For both disks and tapes, it will show the number of devices, average write amount, max write amount, average read amount, and max read amount, separated by type.

System Operations - Status and Alerts:

- Server Status: Buttons for each server color coded to server status, with drop down details on hover.
- Devices and Drives Status: For both tapes and disks, show which devices or drives are locked, along with mover and PVR counts.
- Active Storage Classes: Shows which disk and tape storage classes are approaching storage thresholds, along with the number of each going through migrate and purge operations.
- RTM Summary: Show the number of read, write, or migrate RTM jobs, the number of RTM jobs that are in progress or blocked, and the durations of ongoing RTM jobs.



Below those two rows, there is also a **Latest Alarms and Warnings** row. The alarms and events table starts collapsed, and show at most the latest 2000 events. It is also filterable to show only events of "Critical" and/or "Warning" severity.

14.3. Quick startup of the SSM CLI

This quick setup chapter provides the steps to get started with the SSM Command Line Interface (ssmcli(7)).

14.3.1. Install or Update the SSM CLI

The ssmcli(7) comes bundled with the HPSS system. However, it's recommended that users needing access to the SSM from the CLI should do so from their own computer. The ssmcli(7) is designed to be platform-agnostic and can be installed and run from the Python interpreter on the user's

computer.

This package must be copied to the user's computer and installed using Python's pip command. The ssmcli(7) wheel file can be obtained from the `/opt/hpss/lib/dist` directory from the hpss-core RPM and installed via the Python `pip` tool.

```
pip install ssmcli-X.Y-py3-none-any.whl
```

The X.Y part denotes the version of HPSS the ssmcli(7) is built against.



On Windows, macOS, and RHEL/Debian-based systems, Python may be installed differently. In some cases, Python and pip must be installed separately (for example, via the Microsoft Store on Windows). Additionally, the commands may use version-specific names such as `python3` and `pip3` instead of `python` and `pip`.

To update (upgrade) the ssmcli(7), use the `--upgrade` flag:

```
pip install --upgrade ssmcli-X.Y-py3-none-any.whl
```

14.3.2. Configuring the SSM CLI

This section will explain how to configure the ssmcli(7) with the settings required to interact with HPSS. These settings can be located in multiple places, such as user environment variables, a local configuration file, or a command-line override parameter. The general order of precedence is as follows: **command-line options** override **environment variables**, which override the **configuration file**.

- Command-line Option for the SSM CLI - Options such as --output, --api-url, and --verbose parameters.
- Environment Variables for the SSM CLI - Options stored in the user's environment variables.
- Configuration settings for the SSM CLI - The `config.yml` file, updated when running the `ssmcli configure` command.

The preferred method of any configuration change is through the `configure` command. This command provides a consistent interface for modifying system settings and ensures that configuration updates are validated and applied correctly.

Running the `configure` command without any options or parameters will prompt the user for configuration values. Default or current values will be shown, and pressing Enter without making any changes will keep the existing value:

```
$ ssmcli configure
SSM Secret Access Key [*****]:
SSM API Gateway URL [https://api.example.com/v1]:
Default output format [table]:
```

The fields prompted are required for an SSM command to successfully run. See Creating API Access Tokens for details on creating and using an SSM API Access key.

Additional configuration values that are not typically modified can be set with:

```
$ ssmcli configure set <FIELD> <NEW_VALUE>
```

To see all configuration settings being used by the ssmcli(7) during runtime:

```
$ ssmcli configure list
```

Or, to see individual configuration settings:

```
$ ssmcli configure get <FIELD>
```

See the ssmcli(7) man page for an extensive list of configuration settings.

14.3.3. Configuration settings for the SSM CLI

On startup, the ssmcli(7) will attempt to create the `config.yml` file in the user configuration directory. This behavior is largely based on the XDG Base Directory Specification on Linux and the official guidelines on macOS and Windows. Below is a practical reference table showing where the `config.yml` file will be stored relative to the platform.

Operating System	Directory	Notes
GNU/Linux	`/.config/ssmcli`	Standard per the XDG Base Directory spec.
macOS	`~/Library/Application Support/ssmcli` (sometimes `~/Library/Preferences`)	macOS separates structured app data and preference files
Windows	`C:\Users\<user>\AppData\Roaming\ssmcli`	Per-user application configuration that should roam with profiles



You can specify a non-default location by setting the `SSM_CLI_CONFIG_FILE` environment variables to another local path. See Environment Variables for the SSM CLI for details.

The format of the `config.yml` file is YAML. The configuration file therefore consists of key-value pairs, lists, and nested sections defined through indentation.

It's generally not recommended to make any changes to this file, as they can easily be overridden by the `configure` command.

14.3.4. Environment Variables for the SSM CLI

Environment variables can be used to configure the behavior of the ssmcli(7) as a convenient way to supply configuration values without command-line arguments or storing information in a configuration file. The SSM CLI reads these variables from the environment at runtime.



Specifying an option with an environment variable will override any value loaded from the configuration file. Specifying a command-line parameter will override the environment variable. See [command-line-options] for more details.

The following example will show you how to configure an SSM CLI environment variable:

On GNU/Linux or macOS:

```
$ export SSM_CLI_OUTPUT=json
```

On Windows Command Prompt:

```
C:\> set SSM_CLI_OUTPUT json
```



With setx, the environment variables changes the value used in all command prompt sessions. Using set will change an environment variable until the end of a command prompt session.

On PowerShell

```
PS C:\> $Env:SSM_CLI_OUTPUT="json"
```

The following environment variables are used by the ssmcli(7):

Environment Variable	Functionality
SSM_CLI_CONFIG_FILE	Specifies the location of the file that the ssmcli(7) uses to store the configuration.
SSM_CLI_API_URL	Specifies the API URL that is used for all SSM API requests.
SSM_CLI_LOG_PATH	Specifies the log file output the of the ssmcli(7)
SSM_CLI_OUTPUT	Override the configured output formatting.
SSM_CLI_VERBOSE	Specifies whether verbose logging should be enabled.

14.3.5. Command-line Option for the SSM CLI

Command-line options can be used as global parameters to override configuration settings in the

ssmcli(7) for the single command at runtime.

An example which may be commonly used:

```
$ ssmcli --output=json file-family list
{ ... }
```

Each option can use either a space or equals sign seperating the argument from the option name.

The following command-line options are used by the ssmcli(7):

Environment Variable	Functionality
--config-file	Specifies the location of the file that the ssmcli(7) uses to store the configuration.
--api-url	Specifies the API URL that is used for all SSM API requests.
--log-file	Specifies the log file output the of the ssmcli(7)
--output	Override the configured output formatting.
--verbose/--no-verbose	Specifies whether verbose logging should be enabled.

14.3.6. HTTP configuration for the SSM CLI

The ssmcli(7) provides several optioins for configuring HTTP behavior when communicating with the SSM API. These settings allow you to control TLS verification, specify custom certificate authorities, configure request timeouts, and route requests through HTTP proxies.

These options are useful in environments with strict network policies, corporate proxies, or custom certificate infrastructures.

The following capailities are supported:

- **Disabling TLS certificate verification** - In environments using self-signed certificates or during testing, TLS certificate verification can be disabled. Disabling verification reduces security and should only be used when necessary.
- **Providing a custom CA bundle** - If your organization uses an internal certificate authority, you can provide a CA bundle so the CLI can properly verify server certificates.
- **Configuring request timeouts** - Timeouts can be adjusted to control how long the CLI waits for a connection or response before failing.
- **Using HTTP or HTTPS proxies** - Requests can be routed through a proxy server using standard proxy configuration.

Each setting can be typically configured through the `configure` command with the `http.` using dot notation:

```
$ ssmcli configure set http.verify_ssl false
$ ssmcli configure get http.timeout_connect
30.0
```

14.3.7. Auto-Completion in the SSM CLI

The ssmcli(7) provides a shell-agnostic auto-completion feature that enables the `Tab` key to complete a partially entered command or parameter.

The ssmcli(7) will detect the shell from environment variables such as `SHELL`, `COMSPEC`, or explicit arguments provided in the `--install-completion` command.

The following command will install the auto-completion feature into your shell:

```
$ ssmcli --install-completion
```

The `--install-completion` command writes the completion script into the correct shell configuration location.

Shell	Installation Location
Bash	`~/.bashrc` or `~/.bash_completion`
Zsh	`~/.zshrc`
Fish	`~/.config/fish/completions/ssmcli.fish`
PowerShell	`$PROFILE`

The `--show-completion` will show the auto-completion script for the current shell:

```
$ ssmcli --show-completion
#compdef ssmcli

_ssmcli_completion() {
  eval $(env _TYPER_COMPLETE_ARGS="${words[1,$CURRENT]}" _SSMCLI_COMPLETE=complete_zsh
ssmcli)
}

compdef _ssmcli_completion ssmcli
```

After enable auto-completion, reload your shell, enter a partial command, and press `Tab` to see the available commands:

```
$ ssmcli f:TAB:
file-family       -- Show commands for file families.
```

14.4. Quick startup of hpssgui

We recommend that hpssgui(7) sessions be invoked from the user's desktop computer instead of on the HPSS server machine. hpssgui(7) is an application designed to run in the Java environment on the user's desktop computer and to communicate with the remote SSM System Manager. If hpssgui(7) is executed on the remote System Manager host, it must run through an X Window session and it may run very slowly in that environment. This is a limitation of Java and networks.

We recognize the value of using the remote X functionality as a quick way to get SSM running, but once your system is up, it is highly recommended that you configure local desktop SSM hpssgui(7) clients for all HPSS administrators and operators. Local desktop hpssgui(7) configuration is detailed in Configuration and startup of **hpssgui** and **hpssadm** below.

The following are steps for quickly configuring and starting an SSM GUI client:

1. Use the hpssuser(7) utility to create an SSM user with administrator authority. See The **hpssuser** utility for more information.
2. The **mkhpss** utility generates the `ssm.conf` SSM configuration text file when configuring the SM. See the Configure HPSS infrastructure for more details. Verify the existence of the file `$HPSS_PATH_SSM/ssm.conf`.
3. Start the hpssgui(7) script:

   ```
   % /opt/hpss/bin/hpssgui.pl
   ```

   - Note that the **-m** option can be used to specify the desired SSM configuration file to be used. When this option is not specified, `hpssgui.pl` looks for the `ssm.conf` configuration file in the current directory, then in the directory defined by the HPSS_PATH_SSM environment variable (usually `/var/hpss/ssm`). If the script doesn't find a configuration file in either directory, it will use default values to start the client.
   - Note that the **-d** (debug) and **-S** (log file name) options can be used to capture all levels of hpssgui(7) logging in a text file. Bear in mind, however, that this can generate significant amounts of log data. (See the hpssgui(7) man page.)
   - When you have decided on the hpssgui(7) command line that is best for your installation, it will probably be useful to put the command in a shell script for the convenience of all SSM administrators and operators. For example, create a file called "gui" and put the following in it:

     ```
     /opt/hpss/bin/hpssgui.pl \
     -m /my_directory/my_ssm.conf \
     -d \
     -S /tmp/hpssguiSessionLog.$(whoami)
     ```

Refer to the hpssgui(7) man page for an extensive list of command line options. For example, some sites prefer to set the date format to a USA military format using the **-D "kk:mm:ss dd-MMM-yyyy" option**. Additionally, SSM configuration file below provides a table of variables you can set in the

SSM configuration file instead of using command line options; this section also covers all the various files that the hpssgui(7) script uses.

14.5. Configuration and startup of hpssgui and hpssadm

This section describes in detail the procedures for configuring SSM and creating an SSM user account with the proper permissions to start up an hpssgui(7) or hpssadm(7) session. It also explains how to install the SSM client on the user's desktop (the recommended configuration for hpssgui(7)) and how to deal with special situations such as firewalls.

In the discussion that follows, *authentication* ensures that a user is who they claim to be relative to the system. *Authorization* defines the user's rights and permissions within the system.

Like other components of HPSS, SSM authenticates its users by using either Kerberos or UNIX. Users of the hpssgui(7) and hpssadm(7) utilities are authenticated to SSM by either a Kerberos principal and a password or by a UNIX username and a password. The System Manager must be configured to use the appropriate authentication and a Kerberos principal or UNIX user account must be created for each SSM user.

Unlike other components of HPSS, SSM does not use LDAP or UNIX to authorize its users. SSM users are authenticated based on their entries in the HPSS DB2 AUTHZACL table. Through this table, SSM supports two levels of SSM client authorization:

admin This security level is normally assigned to an HPSS administrator. The admin user can view all SSM windows and perform all control functions provided by SSM.

operator This security level is normally assigned to an HPSS operator. The operator user can view most SSM windows and perform all SSM control functions except for changing the HPSS configuration.

Configuration of an SSM user requires that:

1. The System Manager is configured to accept the desired authentication mechanism.
2. The proper user accounts are created:
   - UNIX or Kerberos accounts are created for the user authentication.
   - The proper authorization entries for the user are created in the AUTHZACL table.
3. The proper SSM configuration files are created and installed.

See Configuring the System Manager authentication for SSM clients, Creating the SSM user accounts and SSM configuration file for the procedures for these tasks.

See SSM help files (optional) for instructions on installing the SSM help package.

See SSM desktop client packaging for instructions for installing hpssgui(7) or hpssadm(7) on the user's desktop.

See Using SSM through a firewall for advice about using SSM clients through a network firewall.

14.5.1. Configuring the System Manager authentication for SSM clients

The System Manager is configured initially by **mkhpss** for new HPSS systems or by the conversion utilities for upgraded HPSS systems to use the proper authentication mechanism.

If it is necessary later to modify the authentication mechanism for hpssgui(7) or hpssadm(7) users, or to add an additional mechanism, bring up the *Servers* window, select the System Manager, and click the **Configure** button. On the *System Manager Configuration* window, select the Interface Controls tab. For the SSM Client Interface, make certain the check box for the desired authentication mechanism, KRB5 or UNIX, is selected. Both mechanisms may be enabled if desired.

Next, select the Security Controls tab. Make certain one of the authentication service configurations is set to use the desired **Mechanism**, an **Authenticator Type** of Keytab, and a valid keytab file name for **Authenticator** (default is `/var/hpss/etc/hpss.keytab` for Kerberos, or `/var/hpss/etc/hpss.unix.keytab` for UNIX).

To remove an authentication mechanism from the System Manager, so that no SSM user may be authenticated using that mechanism, reverse the above process. Unselect the mechanism to be removed from the SSM Client Interface on the Interface Controls tab. On the Security Controls tab, change the **Mechanism** and **Authenticator Type** fields of the mechanism to be removed to Not Configured, and change **Authenticator** to blank.

See Interface controls and Security controls for more information.

14.5.2. Creating the SSM user accounts

14.5.2.1. The hpssuser utility

The hpssuser(7) utility is the preferred method for creating, modifying or deleting SSM users. It creates the necessary UNIX or Kerberos accounts. It creates an entry in the AUTHZACL table for the user with the proper authorization.

The following is an example of using the hpssuser(7) utility to provide administrative access to SSM to user "john". In this example, the user already has either a UNIX or Kerberos account.

```
% /opt/hpss/bin/hpssuser -add john -ssm
[ adding ssm user ]
1) admin
2) operator
Choose SSM security level
(type a number or RETURN to cancel):
> 1
[ ssm user added : admin ]
```

After SSM users are added, removed, or modified, the System Manager will automatically discover the change when the user attempts to log in. See the hpssuser(7) man page for details.



Removing an SSM user or modifying an SSM user's security level won't take effect until that user attempts to start a new session. This means that if an SSM user is

removed, any existing SSM sessions for that user will continue to work; access won't be denied until the SSM user attempts to start a new SSM session. Likewise, if the SSM user's security level is changed, any existing sessions for that user will continue to work at the old security level; the new security level access won't be recognized until the SSM user starts a new SSM session.

14.5.2.2. SSM user authorization

SSM user authorization is set properly by the hpssuser(7) utility with no further modification required. This section explains how the authorization levels are stored internally and how they may be viewed for debugging or modified.

The SSM admin and operator security authorization levels are defined in the AUTHZACL table in the HPSS DB2 database. Each SSM user must have an entry in this table. The permissions supported in the table are:

- **r** indicating *read*
- **w** indicating *write*
- **x** indicating *execute*
- **c** indicating *control*
- **i** indicating *insert*
- **d** indicating *delete*
- **t** indicating *test*

SSM administrators must be granted all permissions: `rwxcidt`. SSM operators must be granted `r-c-t` permissions. All other permission combinations are not recognized by the SSM server and will be treated as no permissions at all.

The AUTHZACL table may be viewed or updated with the hpss_server_acl(7) utility. The hpssuser(7) utility program creates and deletes SSM user entries in the AUTHZACL table using the hpss_server_acl(7) utility. Normally, there is no need to invoke the hpss_server_acl(7) utility directly because it is invoked by the hpssuser(7) utility. However, it is a useful tool for examining and modifying the authorization table.



Access to the hpss_server_acl(7) program, hpssuser(7) program, HPSS DB2 database, and all HPSS utility programs should be closely guarded. An operator with permission to run these tools could modify the type of authority granted to anyone by SSM. Note that access to the database by many of these tools is controlled by the permissions on the `/var/hpss/etc/mm.keytab` file.

Here is an example of using the hpss_server_acl(7) utility to set up a client's permissions to be used when communicating with the SSM server. Note that the **default** command should be used only when creating the ACL for the first time, as it removes any previous entries for that server and resets all the server's entries to the default values:

```
% /opt/hpss/bin/hpss_server_acl
```

```
hsa> acl -t SSM -T ssmclient
hsa> show
hsa> default # Note: ONLY if creating acl for the first time
hsa> add user <username> <permissions>
hsa> show
hsa> quit
```

If the ACL already exists, this command sequence gives user "bill" operator access:

```
% /opt/hpss/bin/hpss_server_acl
hsa> acl -t SSM -T ssmclient
hsa> show
hsa> add user bill r--c--t
hsa> show
hsa> quit
```



Removing an SSM user or modifying an SSM user's security level won't take effect until that user attempts to start a new session. This means that if an SSM user is removed, any existing SSM sessions for that user will continue to work; access won't be denied until the SSM user attempts to start a new SSM session. Likewise, if the SSM user's security level is changed, any existing sessions for that user will continue to work at the old security level; the new security level access won't be recognized until the SSM user starts a new SSM session.

14.5.2.3. User keytabs

A keytab is a file containing a user name and an encrypted password. The keytab file can be used by a utility program to perform authentication without human interaction or the need to store a password in plain text. Each user who will run the hpssadm(7) or hpssgui(7) utility may employ a keytab. It is recommended that one keytab file per user be created rather than one keytab containing multiple users.

Each keytab file should be readable only by the user for whom it was created. Each host from which the hpssadm(7) or hpssgui(7) utility is executed must be secure enough to ensure that the user's keytab file cannot be compromised. An illicit process which gained access to a Kerberos keytab file could gain the user's credentials anywhere in the Kerberos realm; one which gained access to a UNIX keytab file could gain the user's credentials at least on the System Manager host. The hpssgui(7) and hpssadm(7) utilities will look for a default keytab in the user's home directory, called `krb5.keytab` for Kerberos or `keytab` for UNIX.

Keytabs are created for the user by the hpssuser(7) utility when the keytab file path is specified. Keytabs may also be created manually with the **hpss_krb5_keytab** or hpss_unix_keytab(7) utility, as described below.

Keytabs for Kerberos authentication: hpss_krb5_keytab

The **hpss_krb5_keytab** utility may be used to generate a keytab with Kerberos authentication in the

form usable by the hpssadm(7) and hpssgui(7) programs. See the **hpss_krb5_keytab** man page for details.

The Kerberos keytab is interpreted by the KDC of the Kerberos realm specified by the hpssadm(7) and hpssgui(7) utilities (see the **-k** and **-u** options on the respective man pages). This must be the same Kerberos realm as that used by the System Manager. This means the **hpss_krb5_keytab** utility must be executed on a host in the same realm as the System Manager.

This example for a user named *joe* creates a Kerberos keytab file named `krb5.keytab`:

```
% /opt/hpss/bin/hpss_krb5_keytab -f /home/joe/krb5.keytab
HPSS_ROOT is not set; using /opt/hpss
KRB5_INSTALL_PATH is not set; using /krb5
password:
Your keytab is stored at /home/joe/krb5.keytab
```

Keytabs for UNIX authentication: hpss_unix_keytab

The hpss_unix_keytab(7) utility may be used to generate a keytab with UNIX authentication in the form usable by the hpssadm(7) and hpssgui(7) programs.

The UNIX keytab is interpreted on the host on which the System Manager runs, not the host on which the hpssadm(7) client utility runs. The encrypted password in the keytab must match the encrypted password in the password file on the System Manager host. Therefore, the hpss_unix_keytab(7) utility must be executed on the host on which the System Manager runs.

The hpss_unix_keytab(7) utility must be able to read the user's encrypted password from the password file.

This example for a user named *joe* creates a UNIX keytab file named `keytab`:

```
% /opt/hpss/bin/hpss_unix_keytab -f /home/joe/keytab add joe
```

This command copies the encrypted password from the password file into the keytab.

Note: The **-r** option of the hpss_unix_keytab(7) utility will first randomize the user's password and then place it into the keytab file. This is rarely desirable. The **-p** option will change the password first, and then place it into the keytab file. If multiple keytab files exist for this user, this option will invalidate the other keytabs for this user, even if the password is the same. Therefore, this option is also rarely desirable.

14.5.3. SSM configuration file

The SSM clients use the SSM configuration file `ssm.conf` for configuration.

The **mkhpss** utility creates the SSM configuration file for the security mechanism supported by SSM. The **mkhpss** utility stores the generated `ssm.conf` at $HPSS_PATH_SSM; the default location is `/var/hpss/ssm`. The configuration file contains host-specific and site-specific variables that the

clients read. The variables contain information about:

- SSM hostname
- SSM RPC number
- SSM RPC protection level
- SSM security mechanism
- SSM UNIX realm (only if using UNIX authentication)

If any of these configuration parameters are modified, the `ssm.conf` file must be updated and redistributed from the server machine to all of the SSM client machines.

Users can also use their SSM configuration file to manage SSM client parameters instead of using the command line options. The clients can be directed to use an alternate SSM configuration file with the **-m** option. The default SSM configuration file contains comments describing each of the available parameters that may be set along with any associated environment variable and command line option. The following table documents these variables and the corresponding command line options:

Table 17. SSM general options

File Option	Command Line Option	Functionality
HPSS_SSM_ALARM_RATE	-A	Alarm refresh rate
LOGIN_CONFIG	-C	Full path to `login.conf` file
HPSS_SSM_DATE_FORMAT	-D	Date format pattern
HPSS_SSM_ALARMS_GET	-G	Number of alarms requested per poll
HPSS_SSM_LIST_RATE	-L	How long hpssgui(7)/hpssadm(7) waits between polling for lists
HPSS_SSM_ALARMS_DISPLAY	-N	Max number of alarms displayed by hpssgui(7)
HPSS_SSM_CLASSPATH	-P	Full path to `hpss.jar` file
LOG_FILE	-S	Full path for session log file
HPSS_SSM_WAIT_TIME	-W	How long SM waits before returning if object is unchanged
HPSS_SSM_CNTRY_CODE	-c	Country code, internationalization
HPSS_SSM_DEBUG	-d	Debug flag
HPSS_SSM_SM_HOST_NAME	-h	System manager hostname
HPSS_SSM_USER_PREF_PATH	-i	Path to SSM preferences
JAVA_BIN	-j	Path to Java bin directory
KRB5_CONFIG	-k	Full path to `krb5.conf` file
HPSS_SSM_LANG_CODE	-l	Language code, internationalization

File Option	Command Line Option	Functionality
SSM_CONFIG	-m	Full path to SSM configuration file
HPSS_SSM_SM_PORT_NUM	-n	port number *or* RPC number:program number
HPSS_SSM_CLIENT_IP	-p	Client IP address
HPSS_SSM_RPC_PROT_LEVEL	-r	RPC protection level
HPSS_SSM_SEC_MECH	-s	Security mechanism
HPSS_SSM_UNIX_REALM	-u	UNIX realm

Table 18. HPSSGUI-specific options

File Option	Command Line Option	Functionality
HPSSGUI_AUTHENTICATOR	-a	Authenticator (keytab path name)
HPSSGUI_LOOK_AND_FEEL	-F	Look and feel
HPSSGUI_MO_RATE	-M	How long hpssgui(7) waits between polling for managed objects
HPSSGUI_METAL_THEME	-T	Theme file, for look and feel
HPSSGUI_METAL_BG	-b	Background color
HPSS_SSM_HELP_FILES_PATH	-f	Path to help files
HPSS_SSM_HELP_URL_TYPE	-g	Help path URL type

Table 19. HPSSADM-specific options

File Option	Command Line Option	Functionality
HPSSADM_USER_NAME	-U	User name for hpssadm(7)
HPSSADM_AUTHENTICATOR	-a	Authenticator (keytab path name)
HPSSADM_BATCH_MODE	-b	Batch mode flag

Information on tuning client polling rates for optimal performance is available in the hpssadm(7) and hpssgui(7) man pages.

Options are specified, in precedence order, by: 1) the command line; 2) the user's environment (see the man pages for environment variable names); 3) the SSM configuration file; or 4) internal default values.

14.5.3.1. login.conf

The `login.conf` file is a login configuration file that specifies the security authentication required for the hpssgui(7) and hpssadm(7) programs. A copy of the `login.conf` file is included in the `hpss.jar` file and should require no site customization. However, a template for the file is provided

in `/opt/hpss/config/templates/login.conf.template` should the site need to customize the security mechanisms.

See the `/opt/hpss/config/templates/login.conf.template` file for details.

14.5.3.2. krb5.conf (for use with Kerberos authentication only)

The `krb5.conf` file is the Kerberos configuration file which allows the client to authenticate to the Kerberos realm. This file is only required if Kerberos authentication is used. The Kerberos installation process generates a default Kerberos configuration file in `/etc/krb5.conf`.

The following is an example of this file. Realm names, hostnames, and the other data must be customized to operate properly in the user's site environment.

`krb5.conf`:

```
[logging]
default = FILE:/var/hpss/log/krb5libs.log
kdc = FILE:/var/hpss/log/krb5kdc.log
admin_server = FILE:/var/hpss/log/kadmind.log

[libdefaults]
ticket_lifetime = 24000
default_realm = EXAMPLE.COM
default_keytab_name = /etc/v5srvtab
default_tkt_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
default_tgs_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96

[realms]
EXAMPLE.COM = {
kdc = example.com:88
admin_server = example.com:749
}

[domain_realm]
example.com = EXAMPLE.COM
```

Note that having strong encryption types can cause authentication failures. Newer Kerberos libraries may define new encryption types that older libraries don't know of. Strong encryption types are known to fail for Java implementations that have the standard JCE policy files installed.

14.5.4. SSM help files (optional)

The SSM help files are an HTML version of the *HPSS Admin Guide*. Individual sections of this guide are available from the Help menus on the SSM windows.

To access help windows from the hpssgui(7), the help files must be accessible from each client machine. We recommend storing these files in a file system shared by the clients so that they don't need to be installed on every SSM client machine. By default, the hpssgui(7) script looks for the help

files in $HPSS_HELP_FILES_PATH. The default location is `/opt/hpss/ssmhelp` and can be overridden by using the **-f** option.

Help files are distributed with HPSS or can be downloaded from the HPSS web site. They should be installed in the $HPSS_HELP_FILES_PATH location and/or the path specified by the **-f** option. Refer to the Locate HPSS documentation and set up manual pages section for instructions on how to install the help files. See the hpssgui(7) man page for more details.

14.5.5. SSM desktop client packaging

A full installation of HPSS is not needed on machines used only for executing hpssgui(7) or hpssadm(7). These machines, referred to here as "SSM client machines", only require the proper version of Java plus a subset of HPSS components.

It is strongly recommended that a desktop configuration be created and installed for each hpssgui(7) user. The hpssgui(7) program may run very slowly if it is executed on the System Manager machine and displayed back to the user's desktop via remote X.

There is no advantage to executing the hpssadm(7) program on the desktop machine. It will perform just as well when executed remotely as on the desktop. In fact, it is recommended that hpssadm(7) be executed on the System Manager machine rather than on the user's desktop since this simplifies the dissemination and protection of the user keytabs. Nonetheless, instructions are included here for packaging the hpssadm(7) for sites who have a need to execute it on the desktop.

If the SSM code on the System Manager machine is recompiled, or if the System Manager is reconfigured, the client package will no longer work since it is then possible the `hpss.jar` file may be out of sync with the System Manager code. Since each client will have its own copy of the `hpss.jar` file, the `hpss.jar` file should be redistributed to each client. This can be done by redistributing the entire SSM client package or by just redistributing the `hpss.jar` file.

Automatic SSM client packaging and installation describes how to use the hpssuser(7) utility to package these components. Manual SSM client packaging and installation describes how to select and package the components manually.

14.5.5.1. Automatic SSM client packaging and installation

The hpssuser(7) utility provides a mechanism for packaging all the necessary client files required to execute the hpssgui(7) program on the user's desktop host. Refer to the hpssuser(7) man page for more information on generating an SSM Client Package. These files may also be copied manually; see Manual SSM client packaging and installation for a list of the required files.

This example creates an SSM Client Package named "ssmclient.tar":

```
%/opt/hpss/bin/hpssuser -ssmclientpkg ssmclient.tar
[ packaging ssm client ]
[ creating ssmclient.tar ]
hpssgui.pl
hpssgui.vbs
hpss.jar
krb5.conf
```

```
ssm.conf
[ packaged ssm client in ssmclient.tar ]
```

Once the SSM Client Package has been generated simply FTP the tar file over to the client node and then extract the member files to the desired location.

14.5.5.2. Manual SSM client packaging and installation

This section describes the manual installation of the necessary client files required to execute the hpssgui(7) or hpssadm(7) program on the user's desktop host. The hpssuser(7) utility also provides a mechanism for packaging these files automatically; see Automatic SSM client packaging and installation.

The desktop machine requires the proper version of Java and the following HPSS files, which should be copied from the host on which the SSM System Manager executes:

- scripts: `hpssgui.pl`, `hpssgui.vbs`, `hpssadm.pl`, or `hpssadm.vbs`
- `hpss.jar`
- `ssm.conf`
- `krb5.conf` (if using Kerberos authentication)
- user keytab (if using hpssadm(7))
- help files (optional)

These are the default locations of these files on the SSM System Manager host, from which they may be copied:

```
startup scripts   /opt/hpss/bin
hpss.jar          /opt/hpss/bin
ssm.conf          /var/hpss/ssm
krb5.conf         /etc/krb5.conf
keytab file       /var/hpss/ssm/keytab.USERNAME
help files        /opt/hpss/ssmhelp
```

These files may be installed in any location on the SSM client machines. The user must have at least read access to the files.

The SSM startup scripts `hpssgui.pl`, `hpssgui.vbs`, `hpssadm.pl`, and `hpssadm.vbs` provide the user with a command line mechanism for starting the SSM client. The `hpssgui.pl` script is a Perl script for starting the SSM Graphical User Interface and the `hpssadm.pl` script is a Perl script for starting the SSM Command Line User Interface. These scripts work on Linux, or Windows platforms so long as Perl is installed on the host. The `hpssgui.vbs` script is a Visual Basic script for starting the Graphical User Interface and the `hpssadm.vbs` script is a Visual Basic script for starting the SSM Command Line User Interface. These scripts work only on Windows platforms.

These scripts depend on the ability to read the other files in the package. See the hpssgui(7) and hpssadm(7) man pages for details.

The `hpss.jar` file contains the hpssadm(7) and hpssgui(7) program files. This is stored on the server machine under $HPSS_PATH_BIN; the default location is `/opt/hpss/bin`. If the SSM source code on the server machine is recompiled or upgraded, the `hpss.jar` file must be redistributed to all of the SSM client machines.

The keytab is used only by the hpssadm(7) program. See User keytabs for details. See SSM help files (optional) for a description of the Help Files.

A writable directory is required for hpssgui(7) or hpssadm(7) session logs, if these are desired. The session log is an ASCII file that stores messages generated by the hpssadm(7) or hpssgui(7) programs. By default, the hpssgui(7) and hpssadm(7) scripts do not create session logs, but it is strongly encouraged that this capability be enabled by using the **-S <location>** option when running the script. The recommended location is `/tmp` on UNIX-like systems or `c:\tmp` on Windows systems. See the hpssgui(7) and hpssadm(7) man pages for more information on creating a session log. Having the session log available helps when debugging problems with the SSM client applications. It is the first thing that the SSM developers will ask for when someone is having problems with the hpssgui(7) or hpssadm(7).

14.5.6. Using SSM through a firewall

14.5.6.1. The firewall problem

SSM clients require the use of several network ports which may be blocked if the client and System Manager are on opposite sides of a network firewall. Up to three ports may be affected:

- SSM clients must be able to access the port upon which the System Manager listens for requests.
- If the System Manager follows the default behavior of letting the portmapper select this port, then SSM clients also need access to port 111 in order to ask the portmapper where the System Manager is listening.
- If Kerberos authentication is used, then SSM clients additionally need access to port 88 to communicate with the KDC.

14.5.6.2. Solutions for operating through a firewall

SSM can operate through a firewall in three different ways:

- The SSM clients can use ports exempted by the network administrator as firewall exceptions. See the **-n** option described in the hpssgui(7) and hpssadm(7) man pages.
- The hpssgui(7) and hpssadm(7) can contact the System Manager across a Virtual Private Network (VPN) connection. See the **-p** and **-h** options described in the hpssgui(7) and hpssadm(7) man pages.
- The SSM clients can contact the System Manager across an SSH tunnel. See the instructions for tunneling in the hpssgui(7) man page.

The firewall exception is the simplest of these. However, security organizations are not always willing to grant exceptions.

The VPN option is usually simple, regardless of how many ports are needed, but requires the site to

support VPN. The site must also allow the VPN users access to the ports listed in The firewall problem, but not all sites do.

The SSH tunneling option has the advantage that it can be used almost anywhere at no cost. It has the disadvantage that the tunnel essentially creates its own firewall exception. Some security organizations would rather know about any applications coming through the firewall and what ports they are using rather than have users create exceptions themselves without the awareness of security personnel. A second disadvantage of tunneling is that if a particular client machine is compromised, any tunnels open on that client could also be compromised. The client machine may become a point of vulnerability and access to the other machines behind the firewall. A third disadvantage is that tunneling can be complex to set up, requiring slight or significant variations at every site.

The firewall and tunneling options both benefit from reducing the number of ports required:

- The need for port 111 can be eliminated by making the System Manager listen on a fixed port. To do this, set the HPSS_SSM_SERVER_LISTEN_PORT variable in the `env.conf` file to the desired port and restart the System Manager. Then use the **-n** option with the hpssgui(7) and hpssadm(7) startup scripts to specify this port.
- The need for port 88 can be eliminated only by avoiding Kerberos and using UNIX authentication.
- There is no way to eliminate the need for the port on which the System Manager listens.

14.5.6.3. Example: Using hpssgui through a firewall

Here is an example of how a particular site set up their hpssgui(7) SSM client sessions using krb5 authentication outside a firewall. Many of the items are site-specific, so modifications will need to be made to suit each site's specific needs. Where this procedure would differ for a site using UNIX authentication, the UNIX instructions are also included.

At this site, VPN users were not allowed access to all the ports listed in The firewall problem so they had to use a combination of VPN and SSH tunneling.

- Create a directory on the client machine to hold the SSM client files. It is recommended that a separate directory be created for each server hostname that the client will contact.
- Verify that the proper version of Java is installed. Add the Java `bin` directory to the user's $PATH, or use the **-j** switch in the hpssgui(7) script, or set JAVA_BIN in the user's `ssm.conf` file. Java can be downloaded from http://www.java.com.
- Obtain files from the server machine:
  - Obtain the preferred hpssgui(7) script for the client system from `/opt/hpss/bin` on the server machine and place it in the directory created on the client machine (see SSM desktop client packaging).

    There are several script options. Only one version of the script is needed:

    - `hpssgui.pl` which is written in Perl and can be used on any system that has Perl installed. This is true for any major UNIX operating systems as well as Mac OS. For Windows users, Perl must be installed to use this version of the script. Users can easily obtain this from

the web. A good Perl distribution for Windows is available at http://www.activestate.com.

- `hpssgui.vbs` is a Visual Basic Script version for Windows users. This version requires no prerequisite software.

- Obtain the `ssm.conf` file from `/var/hpss/ssm` on the server machine and place it in the directory where the hpssgui(7) script resides. Alternately, specify the file to the hpssgui(7) script with the **-m** option, if desired.
- Obtain the `hpss.jar` file from `/opt/hpss/bin` on the server machine and place it in the directory where the hpssgui(7) script resides. If FTP is used to copy the file, make sure the copy is done in binary mode. If the file is installed in a different directory, specify it to the hpssgui(7) script with the **-P** option, or by using configuration file settings or the appropriate environment variable (see the hpssgui(7) man page).

- If Kerberos authentication is used, be sure to get the `krb5.conf` file that resides on the SSM server. This file should be located at `/etc/krb5.conf`. Place this file on the client machine in the directory where the hpssgui(7) script resides. Alternately, specify this file to the hpssgui(7) script with the **-k** option. Verify that UDP port 88 on the SSM Server machine is accessible; if not, then hpssgui(7) will fail.
- To get access to ports inside the firewall, we can use a VPN connection or one or more SSH tunnels.
  - Using a VPN connection will make it appear that we are inside the firewall. In this case, no tunnels are needed. If the firewall does not permit SSH connections, SSH tunnels cannot be used. Set up the VPN connection on the client machine.
  - If using one or more SSH tunnels is preferred, on the SSM server machine, set the HPSS_SSM_SERVER_LISTEN_PORT variable in the `env.conf` file to a specific port (for example, 49999). Restart the System Manager so that it will recognize this variable.

    On the client machine, set up an SSH tunnel where 49999 corresponds to the HPSS_SSM_SERVER_LISTEN_PORT, the user name is *joe* and the SSM Server machine is "example.com".

    ```
    % ssh -N -f -L 49999:localhost:49999 joe@example.com
    ```

- On the client machine, run the GUI:
  - For Kerberos authentication:

    ```
    % hpssgui.pl -S hpssgui.sessionlog -k krb5.conf -n 49999 -h localhost
    ```

  - For UNIX authentication:

    ```
    % hpssgui.pl -S hpssgui.sessionlog -s unix -u example.com -n 49999 -h
    localhost
    ```

Either the *HPSS Login* window or the SSM main window should open on the client machine for the user. If it doesn't, then retry the last step, running the GUI, using the **-d** option for debug output and the **-S** option to log output to a session log file. This file will provide some information about what is going wrong.

14.6. Multiple SSM sessions

Multiple concurrent sessions of the graphical user interface and command line utility can be executed by the same user with no special modifications. Each session should specify a unique session log file.

14.7. SSM window conventions

This section lists conventions used by SSM and Java, on which SSM is based. The following list does not cover all features of all windows; it only describes the most important points.

- Lists may be sorted by any column. Click on the column header of the desired column to sort the list by the items in that column. The column header will become highlighted and will display an up or down arrow to indicate the direction of the sort. Click the column header a second time to change the direction of the sort.
- List tables have a field that shows the number of displayed and total items in the list in the format "X/Y" where X is the number of items displayed and Y is the total number of items in the list. The field is left-justified under the table. The X and Y values will differ if preferences are set to filter some items out of the list.
- The button panel to the right of the list can be hidden or displayed by clicking the tall, thin button between the list and button panel labeled ||. If the button is clicked when the panel is displayed, the button panel will hide, allowing more space for the list. The button panel may be redisplayed by clicking the || button again.
- Colors and fonts are used consistently from window to window. They may differ from platform to platform because the default Java Look and Feel settings vary between platforms.

  The hpssgui(7) script accepts the following flag parameters in order to control the graphical user interface's look and feel:

  -F "Look and Feel"

  Valid values: `windows`, `mac`, `motif`, `metal`, and `gtk`. Select the Look and Feel that is applicable to the platform on which the graphical user interface is running. Custom Look and Feels are also available at https://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

  -b "background color"

  The only Look and Feel that supports color settings and themes is the metal Look and Feel. The color may be set by using the color name or hexadecimal Red/Green/Blue value. Here are some examples:

```
Name      Hexadecimal value
red       0xff0000
```

```
green     0x00ff00
blue      0x0000ff
cyan      0x00ffff
yellow    0xffff00
magenta   0xff00ff
```

-T "theme file"

The theme setting is only applicable when used with the metal Look and Feel. There are eight color parameters that can be used to customize the look of HPSS windows: three primary colors, three secondary colors, black, and white. The color settings may be stored in a theme file using the following syntax:

```
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.primary1=<color>
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.primary2=<color>
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.primary3=<color>
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.secondary1=<color>
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.secondary2=<color>
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.secondary3=<color>
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.black=<color>
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.white=<color>
```

<color> should be specified using the color name or Red/Green/Blue hexadecimal value (see the example under the **-b** flag above).

If the theme file location is not specified on the command line, the default value used is `${HOME}/hpss-ssm-prefs/DefaultTheme`.

- Buttons may be "disabled" when the current state of the window does not allow an operation to be performed. In this state, a button is visible but its label text is grayed out and clicking it has no effect. The disabled state occurs when the operation is not supported for the selected item or the SSM user does not have sufficient authority to perform the operation
- A "text" field is any field which displays alphanumeric text or numeric data. This does not include "static" text painted on the window background or labels on things like buttons. Text fields may appear as single or multiple lines and they may be "enterable" (the displayed data can be altered) or "non-enterable" (the displayed data cannot be changed directly).
- Non-enterable text fields have gray backgrounds. A particular field may be enterable under one circumstance but non-enterable under another; for example, a server configuration window's Server ID field is enterable during server creation but may not be changed when modifying a pre-existing configuration record. Additionally, a field is non-enterable when the user does not have sufficient authority to modify the field.
- Enterable text fields have white backgrounds. In most cases, when data in the current field is modified and the field loses focus (the cursor leaves the field), a floppy disk icon will be displayed next to the field to give a visual cue that the field has been changed and that the changes have not been saved. When all changes are made, the user can submit the modifications by clicking one of the window's operation buttons.
- Some enterable text fields are wider than they appear. As typing proceeds and the cursor

reaches the right-most edge of the field, the text automatically scrolls to the left and allows further data entry until the actual size of the field has been reached. Scroll back and forth within the field using the left and right cursor keys to view the contents.

- Some text fields which accept integer values can also accept numeric abbreviations such as "KB", "MB", "GB", "TB", or "EB" to specify kilobytes, megabytes, gigabytes, terabytes, or exabytes, respectively. Character case is ignored. For example, entering "1024" will yield the same results as entering "1kb". The entered value must fall with the acceptable numeric ranges for the specified field.
- Some text fields which accept integer values can accept the values in decimal, octal, or hexadecimal form. For these fields, values which begin with an *x* or *0x* will be interpreted as hexadecimal and values which begin with a zero *0* (but not *0x*) will be interpreted as octal. All other values will be interpreted as decimal.
- A combination box is a non-enterable text field inside a button-like box with a small arrow on the right side. Clicking on the box will display a list of items. Selecting a list item will replace the displayed contents of the combination box's text field. Alternately, the list can be dismissed by clicking the mouse anywhere outside of the drop-down list and the displayed contents will remain unchanged.
- A check box is a field containing a box graphic followed by a label. The box may be hollow, ☐, indicating that the item is not selected. It may be filled in, ■, or contain a check mark, ☑, indicating that the item is selected. Clicking on an enterable check box toggles the state of the selected item. When the state of a check box cannot be modified, it will appear gray in color.
- A radio button is a field containing a circle followed by a label. The circle may be hollow, ○, indicating that the item is not selected or may have a solid interior, ●, indicating that the item is selected. Radio buttons are displayed in groups of two or more items. Only one item within the group can be selected; selecting one button in a group will cause all other buttons in the group to become unselected. When the state of the radio buttons cannot be modified, they will appear gray in color.
- An enterable field containing a cursor is said to have "input focus". If an enterable text field has input focus, typing on the keyboard will enter characters into the field.
- Select, cut, copy, and paste operations can be performed on enterable text fields; on non-enterable fields, only select and copy operations can be performed.
- In some cases, modifying a field value or clicking a button causes the action to be performed immediately. A confirmation window will pop up to inform the user that all changes made to the data window will be processed if the user wishes to continue. If the user selects *No* on the confirmation window, the request will not be processed and any field modifications to the window will continue to be displayed. Some examples are changes to the **Administrative State** field, clicking the Gatekeeper's **Read Site Policy** button, and selecting an entry from the **MPS Storage Class Information Control** combination box.

14.8. Common window elements

Certain SSM buttons and toggle boxes have the same behavior on all SSM windows. Descriptions for these common elements are given below and are not repeated for each window:

Time Created by System Manager field

The last time the System Manager created the structure for this window.

Time Updated by System Manager field

The last time the System Manager updated the data for this window.

Time Received by Client field

The last time the SSM client received an update for this window from the System Manager.

Dismiss button

Closes the current SSM window.

Add button

The **Add** button is displayed on configuration windows when a new configuration record is being created. After the configuration fields are appropriately completed, click the **Add** button to save the data and create the new record. When the Add operation is not permitted, the **Add** button will not be displayed or will appear gray in color.

Update button

The **Update** button is displayed on configuration windows when an existing record is being modified. After the configuration's fields have been modified, click the **Update** button to save the modifications. When the update operation is not permitted, the **Update** button will not be displayed or will appear gray in color.

Delete button

The **Delete** button is displayed on configuration windows of existing records. Click the **Delete** button only when the current record is no longer needed and any dependent records have also been deleted. When the Delete operation is not permitted, the **Delete** button will not be displayed or will appear gray in color.

Start Over button

This resets the current values in a configuration window to the values used when the window was first opened.

Start New button

This replaces the contents of the current configuration window with a new configuration of the same type as the one being viewed. The new configuration's initial values will contain defaults.

Clone (partial) button

This replaces the contents of the current window with a new configuration using some of the current configuration's field values.

Clone (full) button

This replaces the contents of the current window with a new configuration using all of the current configuration's field values.

Freeze

A check box that, while checked, suspends the automatic updates made to an SSM window. This

allows reviewing information at the frozen point in time. Unchecking the check box will reactivate normal update behavior.

Refresh button

This requests an immediate update of the displayed information. This can be useful if the user does not wish to wait for an automatic update to occur.

Preferences button and combination box

Edit button

Clicking the **Edit** button opens a configuration window from which certain display characteristics of the parent window can be modified and saved in a preference record. New preference records can be created by saving the preference record with a new name.

Preferences combination box

Click on the Preference combination box to view a list of available preference records used to control the information displayed on the window. The preference record can be modified by either selecting another preference record or by modifying the current preference record. See the **Edit** button description above.

Status Bar

A non-enterable text field along the bottom of all SSM data windows. Status lines display messages concerning the state of the window and the progress of operations started from the window. To view status messages that are longer than the length of the status bar, either stretch the window horizontally or mouse-over the status message to see the full text displayed as a tool tip. Alternately, the user can view status messages in the session log file. When the status bar has had messages written to it, the most recent messages can be viewed in the status bar's tooltip. If there are status messages to view, rolling the mouse over the status bar without clicking gives a tooltip that says, "Click mouse in status bar to view messages". If there are no status messages then the tooltip says, "No status messages". This message stays up for about four seconds or until the user moves the mouse out of the status bar area. To view up to the last 30 messages that have been written to the status bar, click on the status bar. The tooltip that results will show up to the last 30 messages and will remain visible for ten minutes or until the mouse is moved out of the status bar.

File menu

All SSM data windows have the **File** menu. It consists of menu options for controlling the window's location, the user's session, and printing. The **File** menu offers the following options: **Cascade**, **Page Setup**, **Print**, **Close All** or **Close**, **Logoff**, and **Exit**. The **Cascade**, **Close All**, **Logoff**, and **Exit** menu options are only available on the *HPSS Health and Status* window.

Page Setup

SSM uses Java's print facility to create a dialog box enabling the user to enter directions to be sent to the printer. The Page Setup dialog box can be used to specify print media, page orientation, and margin characteristics. The Page Setup dialog box can also be accessed via the Print dialog box (see below). The **Page Setup** menu item is available on all SSM windows.

Print

The Print dialog box is used to set printing options. The print options that are available are

platform dependent and therefore may vary. Some print options that can be configured include selecting a printer, setting page size and margins, and specifying the number of copies and pages to print. The **Print** menu item is available on all SSM windows.

Close

The **Close** menu option is used to close the currently selected window. The **Close** menu item is available on all SSM windows.

Edit menu

The **Edit** menu is located on all SSM data windows. From each **Edit** menu, the user can access cut, copy, and paste functions which enable the user to remove data from text fields or transfer data among them. Editable text fields can be updated. Non-editable text fields can be copied, but not changed. Field labels cannot be copied.

Most windowing systems provide keyboard shortcuts for the cut, copy, and paste commands. A typical set of keyboard shortcuts is Ctrl-C for copy, Ctrl-X for cut, and Ctrl-V for paste, but details may vary from system to system. Cut or copied text can be pasted into other applications using the keyboard shortcuts.

- To delete data from a text field: highlight the characters to be removed and select **Cut**.
- To move data from one field to another: highlight the characters to be moved and select **Cut**. Then position the cursor where the data should be placed and select **Paste**.
- To copy data from one field to another: highlight the characters to be copied and select **Copy**. Then position the cursor where the data should be placed and select **Paste**.

Column View menu

The **Column View** menu only appears on SSM windows that display an SSM table. An entry for each column in the table appears in the drop down list along with a corresponding check box. If the check box is selected, then the column will appear in the window's table; otherwise the column will be hidden. Clicking on the check box will toggle the hidden or viewable state of the column.

Help menu

All SSM windows have a **Help** menu. See Help menu overview below for detailed information on SSM help.

14.9. Help menu overview

The **Help** menu provides access to online help that is pertinent to the window being displayed. The **Help** menu is available on all SSM data windows but is not available on informational windows such as error messages and confirmation windows. After you select **Help**, the menu will expand to list the help topics that are available for the current window. Selection of a window-related help topic will open an HTML file and jump to the corresponding topic section. Selection of the *HPSS Admin Guide* on a help menu will take the user to the table of contents of the admin guide.

The *HPSS Admin Guide*, along with the *Problem diagnosis and resolution* chapter of the *HPSS Error*

Manual, is the main source for diagnosing and solving problems.

In order to access SSM Help, the help files must be installed and accessible to the graphical user interface. For information on obtaining and installing the SSM Help files, see the Software installation packages.

The SSM Help facility uses two environment variables, HPSS_HELP_URL_TYPE and HPSS_HELP_FILES_PATH, to determine the location of the SSM Help files. The HPSS_HELP_URL_TYPE environment variable specifies the type of URL to aid the browser in locating the help files. Valid URL types are "https:", "http:", or "file:". The default value for the HPSS_HELP_URL_TYPE is "file:". The HPSS_HELP_FILES_PATH environment variable specifies the location of the installation directory for the SSM Help files. The SSM Help files must exist in HTML format for the graphical user interface to display them. The default value for the HPSS_HELP_FILES_PATH environment variable is "/opt/hpss/ssmhelp". The values of these environment variables may be overridden.

14.10. Monitor, Operations, and Configure menus overview

The **Monitor**, **Operations**, and **Configure** menus are used by the System Manager to monitor, control, and configure HPSS. They are available only from the *HPSS Health and Status* window. This section provides a brief description for each submenu option listed under the **Monitor**, **Operations**, and **Configure** menus. See related sections for more detailed information about the window that opens after the menu option has been selected.

14.10.1. Monitor menu

The **Monitor** menu contains submenu items that are commonly used by the system administrator to monitor the status of HPSS. The menu is organized with the submenu items that open SSM list windows at the top and the submenu items that open other window types at the bottom.

Alarms & Events

Opens the *Alarms and Events* window which displays HPSS alarm and event messages. Detailed information on each alarm or event can be viewed by selecting an entry from the list and clicking the **Alarm/Event Info** button.

Devices & Drives

Opens the *Devices and Drives* window which displays information on all configured Mover devices and PVL drives. Devices and drives may be created, deleted, viewed, locked, unlocked, dismounted, and marked as repaired from this window.

Tape Drive Quotas

Opens the *Tape Drive Quotas* window which displays the tape drive quota allocations by PVR and tape drive type. The tape drive Recall Limit may be set from this window.

PVL Jobs

Opens the *PVL Job Queue* window which displays all active PVL jobs. Use this window to get more detailed information on a job or to cancel a job.

Servers

Opens the *Servers* window which displays information on all configured HPSS servers. This window can also be used to perform server-related operations such as configuration, startup, shutdown, and viewing server status.

Storage Classes, Active

Opens the *Active Storage Classes* list window which displays information for all storage classes which have been assigned to a Migration/Purge Server and assigned storage resources. Migration, purge, repack, and reclaim operations can be initiated from this window. This window also provides access to information for each storage class such as the managed object information, configuration record, migration policy, and purge policy. The Migration/Purge Server must be running in order for its storage classes to be active and show up in this list.

RTM Summary

Opens the *RTM Summary List* window which displays summary information for the active Real-Time Monitor (RTM) records. RTM records are maintained by the Core, Gatekeeper, and Mover components of HPSS.

Filesets & Junctions

Opens the *Filesets & Junctions List* window which displays information for the filesets and junctions that are configured in the HPSS system. Filesets and junctions can be created and deleted and details for filesets can be viewed from this window.

Tape Requests

This submenu lists the different tape request list window types.

Check-In

Opens the *Check-In Requests* window which displays all current requests for tape check-ins.

Mount

Opens the *Mount Requests* window which displays all current requests for tape mounts.

Accounting Status

Opens the *Subsystem* list window where the **Accounting Status** and **Start Accounting** buttons can be found.

Lookup HPSS Objects

This submenu lists the type of objects which can be looked up by specifying the object's identifying information.

Cartridges & Volumes

Opens the *Lookup Cartridges and Volumes* window allowing identification of a cartridge or volume by name. A choice can then be made between viewing PVL volume information, PVR cartridge information, or Core Server volume information for the specified cartridge or volume.

Files & Directories

Opens the *Lookup Files and Directories* window into which a pathname can be entered. The

pathname can identify a file, directory, or junction. From this window, click the **Show File/Directory** button to display detailed information about the file, directory, or junction.

Objects by SOID

Opens the *Lookup Object by SOID* window where an HPSS object can be specified by entering its HPSS Storage Object ID (HPSS SOID). This window provides access to bitfile and virtual volume information.

SSM Information

This submenu lists the options available for viewing statistics for the System Manager and the user client session.

System Manager Statistics

Opens the *SSM System Manager Statistics* window to view statistics for the System Manager such as the number of RPC calls, notifications, and messages that were processed. **User Session Information** opens the *User Session Information* window to display the user's login name, authority, and statistics regarding the user's session.

14.10.2. Operations menu

Accounting Report

Opens the *Subsystems* list window where a subsystem can be highlighted and the **Start Accounting** button can be selected to obtain an accounting report.

Drive Dismount

Opens the *Devices and Drives* list window where the **Dismount Drive** button is located.

PVL Job Cancellation

Opens the *PVL Job Queue* window from which PVL jobs may be selected and canceled.

Resources

This submenu lists the operations that can be performed on disk and tape cartridges and volumes.

Import Disk Volumes

Opens the *Import Disk Volumes* window where a list of disk volume labels can be entered and an import request can be submitted.

Import Tape Volumes

Opens the *Import Tape Volumes* window where a list of tape volume labels can be entered and an import request can be submitted.

Import Object Store Volumes

Opens the *Import Object Store Volumes* window where a list of object store volume labels can be entered and an import request can be submitted.

Create Disk Resources

Opens the *Create Disk Resources* window where a list of disk volume labels can be entered

and a request to add the disks to a storage class can be submitted.

Create Tape Resources

Opens the *Create Tape Resources* window where a list of tape volume labels can be entered and a request to add the tapes to a storage class can be submitted.

Create Object Store Resources

Opens the *Create Object Store Resources* window where a list of object store volume labels can be entered and a request to add the object store volumes to a storage class can be submitted.

Delete Resources

Opens the *Delete Resources* window allowing deletion of existing tape or disk storage resources.

Export Volumes

Opens the *Export Volumes* window which exports tape cartridges and disk volumes, making them unavailable to HPSS.

Move Cartridges

Opens the *Move Cartridges To New PVR* window allowing ownership of tape cartridges to be transferred between PVRs.

Migrate/Purge Data

Opens the *Active Storage Classes* window. From this window, a storage class may be highlighted and a migration or purge can be started by selecting the corresponding button.

Repack/Reclaim Tapes

Opens the *Active Storage Classes* window where a storage class may be highlighted and the **Repack Volumes** or **Reclaim Volumes** button selected to perform the operation.

Ping System Manager

Selecting this submenu option tests the connectivity between the GUI and the System Manager. If the ping is successful, nothing will happen. If the ping is unsuccessful, an error message will be displayed.

Refresh Restricted Access Table

Selecting this option causes the Core Server to reread the Restrict Access File in order to refresh the restricted access table. See Refresh Restricted Access Table for more information.

Shutdown

This submenu provides a quick way to send a shutdown request to any server other than the Startup Daemon. If you want to shut down a particular server or set of servers, use the **Shutdown** or **Force Halt** buttons on the *Servers* list window. The System Manager cannot be shutdown via the *Servers* list window. The Startup Daemon cannot be shutdown at all using SSM.

All Non-SSM Servers

Sends a shutdown command to all servers other than the System Manager and Startup

Daemon. Note that some servers may take a few minutes to shut down. To restart the servers, select the servers in the *Servers* list window and click the **Start** button.

System Manager

Sends a shutdown command to only the System Manager.

14.10.3. Configure menu



*It is recommended that the system administrator configure an HPSS system traversing the **Configure** menu in top-down order since some configuration items have a dependency on others.*

Subsystems

This opens the *Subsystems* list window where a list of all configured subsystems can be viewed, new subsystems can be configured, or existing subsystems can be deleted. Additionally, from the *Subsystems* list window, accounting statistics can be viewed and reports can be generated.

Policies

This submenu lists the policy types that can be configured for HPSS.

Accounting

Opens the *Accounting Policy* window allowing configuration and management of the accounting policy. Only one accounting policy is allowed.

Logging

Opens the *Logging Policies* list window allowing configuration and management of the logging policies.

Migration

Opens the *Migration Policies* list window allowing configuration and management of the migration policies.

Purge

Opens the *Purge Policies* list window allowing configuration and management of the purge policies.

Storage Space

This submenu lists the storage space configurations required for HPSS. Classes of Service contain a hierarchy, and hierarchies are made up of a list of storage classes.

Storage Classes

Opens the *Configured Storage Classes* list window allowing configuration and management of storage classes.

Hierarchies

Opens the *Hierarchies* list window allowing configuration and management of storage hierarchies.

Classes of Service

Opens the *Class of Service* list window allowing configuration and management of the Classes of Service.

Servers

This opens the *Servers* list window, which will facilitate server configuration and management.

Global

This opens the *Global Configuration* window allowing the configuration and management of the HPSS global configuration record. Only one global configuration is allowed.

Devices & Drives

This opens the *Devices and Drives* list window allowing configuration and management of devices and drives for use with HPSS.

File Families

This opens the *File Families* list window allowing the configuration and management of file families.

Object Store Accounts

This submenu contains items used to configure accounting information for object store storage classes.

Accounts

Opens the *Object Store Accounts* list window allowing configuration and management of object store accounts.

Identities

Opens the *Object Store Account Identities* list window allowing configuration and management of object store account identities.

Buckets

Opens the *Object Store Account Buckets* list window allowing configuration and management of object store account buckets.

Mappings

Opens the *Object Store Account Mappings* list window allowing configuration and management of object store account mappings.

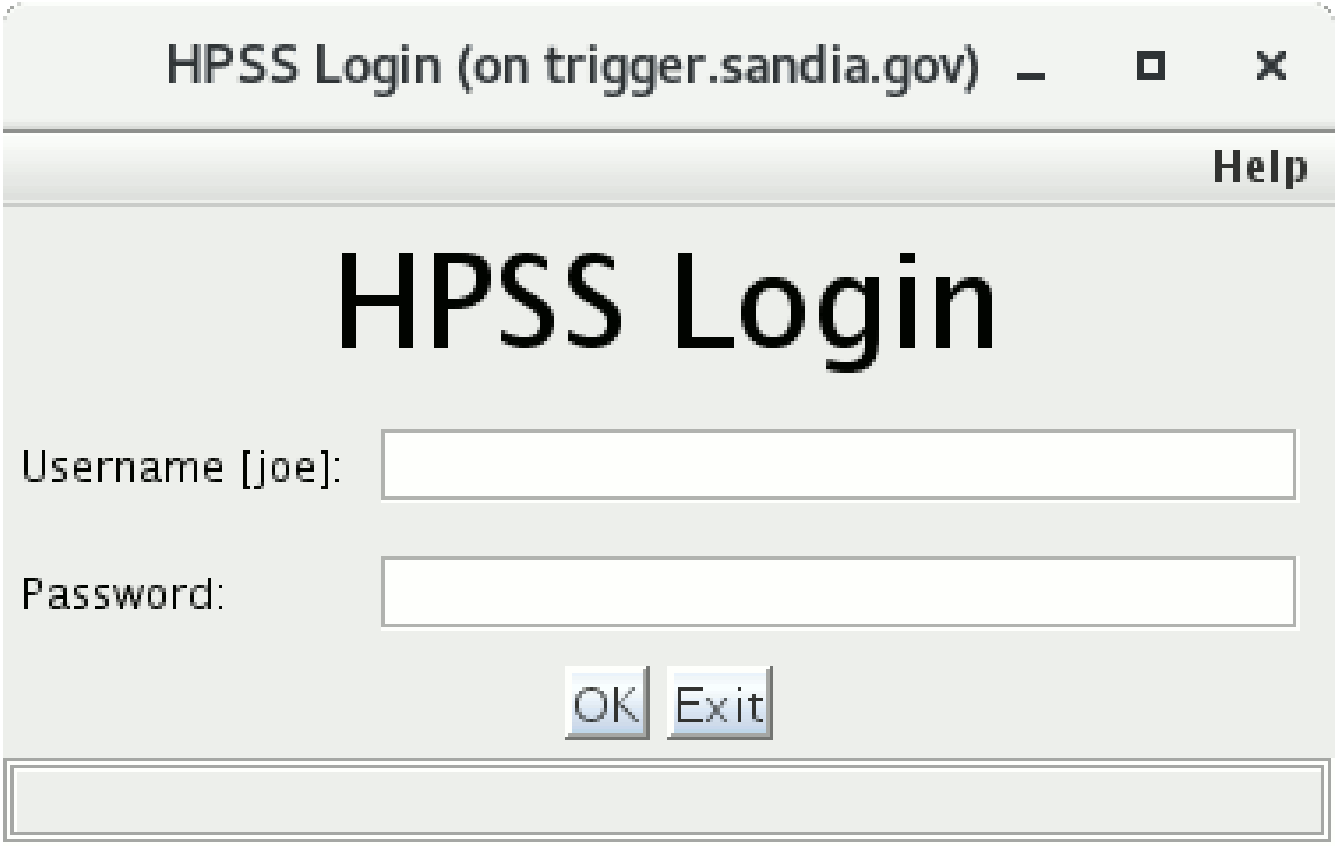
List Preferences

This submenu contains an entry for each SSM list window. A preference record can be created or the default preference record can be modified to allow the user to customize each SSM list window's data view by using filtering methods. See SSM list preferences for more information.

14.11. SSM-specific windows

This section describes the *HPSS Login* window, the *HPSS Health and Status* window, and the SSM information windows.

14.11.1. HPSS Login



The *HPSS Login* window may appear after starting the hpssgui(7) script. The user must supply a valid HPSS user name and password in order to access SSM and monitor HPSS. The SSM GUI will attempt to use existing credentials (for example, Kerberos with kinit) first, a keytab next, and falling back to a username/password. If the login window appears, it means that the SSM GUI did not find valid credentials from the first two methods.

If a login attempt is unsuccessful, review the user session log for an indication of the problem. See the hpssadm(7) or hpssgui(7) man pages for more information about the user session log.

WebSSM and SSMAPI log their own requests and errors in their respective log files.

Field descriptions

User ID

If this field appears, it will be in a form dictated by the `login.conf` mechanism. Enter a valid user ID here.

Password

Some variation of this prompt will appear, as dictated by the `login.conf` mechanism. Enter the password for the user ID.

Note that one or both of these prompts may appear. For instance, the username may be requested first, followed by another window asking for the password. The particulars are up to the mechanism specified in `login.conf`.

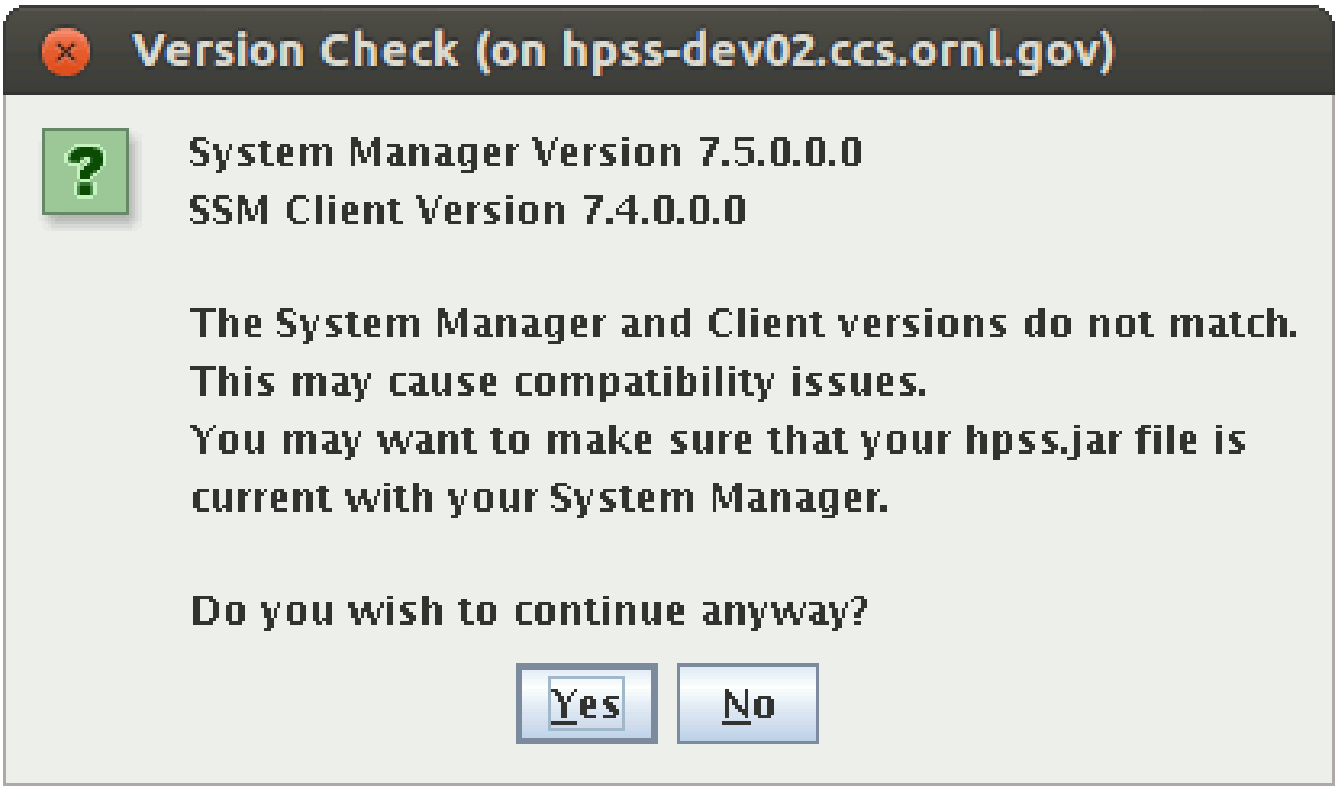
OK

Attempt to contact the System Manager and authenticate the user. If the attempt is successful, the *HPSS Health and Status* window will open. Otherwise, an error message will be displayed in the status bar.

Exit

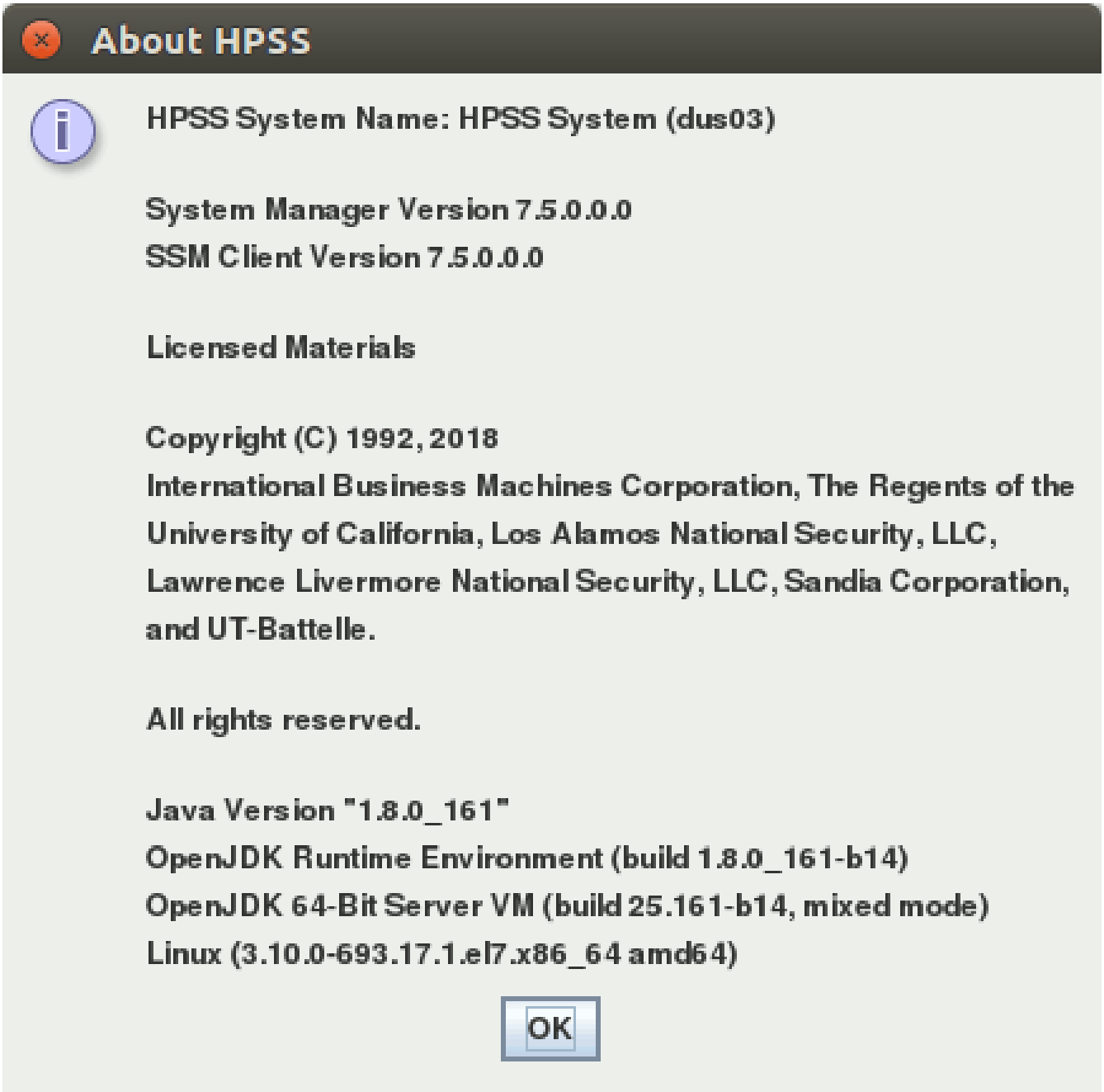
Close the *HPSS Login* window and terminate the hpssgui(7) client session.

If the System Manager and SSM Client versions do not match then the following window will be displayed after the user logs in:



The user may choose to continue logging in or to exit. However, as the dialog says, running with mismatched versions may cause compatibility problems.

14.11.2. About HPSS

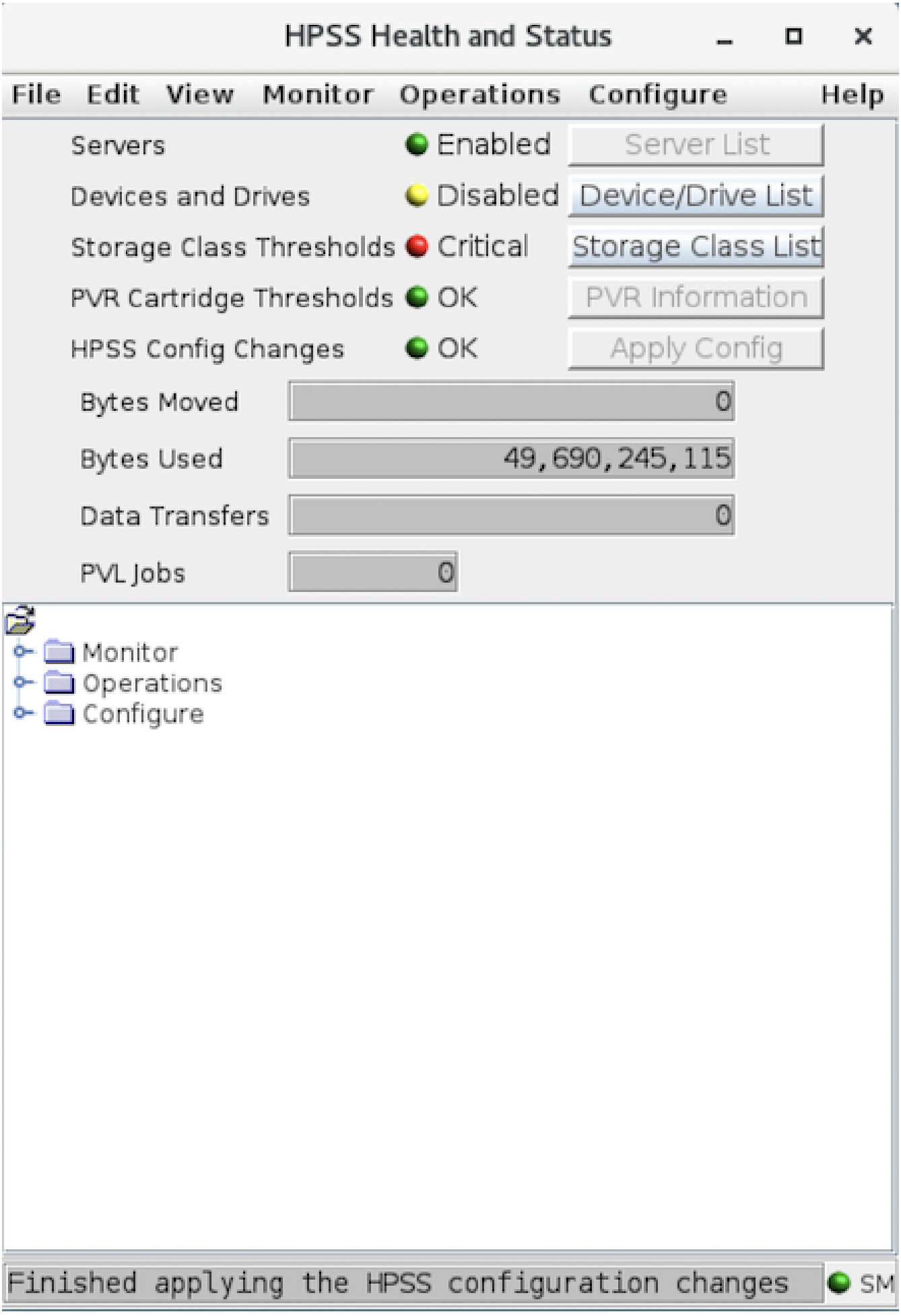


The *About HPSS* window displays version information and a portion of the HPSS copyright statement. The *About HPSS* window is accessible by selecting the Help menu's "About HPSS" submenu from any of the hpssgui(7) windows.

The HPSS System Name and System Manager Version are not displayed when the *About HPSS* window is requested from the *HPSS Login* window. These two pieces of information are not available until the user actually logs into the System Manager.

Differences in the System Manager and SSM Client versions may indicate that the code for the client or System Manager (or both) should be updated.

14.11.3. HPSS Health and Status



When a user successfully connects to the System Manager through the *HPSS Login* window, the

HPSS Health and Status window replaces the *HPSS Login* window on the screen. The *HPSS Health and Status* window will remain on the screen until the user exits or logs out. It provides the main menu and displays information about the overall status of HPSS.

The *HPSS Health and Status* window is composed of several high-level components, each of which is discussed in its own section below.

14.11.3.1. SM server connection status indicator

The SM connection status indicator is located in the bottom right corner of the status bar. When the SM icon is red, the client's connection to the System Manager is lost; when it is green, the connection is active.

14.11.3.2. HPSS status

On the upper section of the *HPSS Health and Status* window are five status fields that represent the aggregate status of the HPSS system:

Servers

This displays the most severe status reported to SSM by any HPSS server.

Devices and Drives

This displays the most severe status as reported to SSM for any configured Mover device or PVL drive.

Storage Class Thresholds

This displays the most severe status reported by the MPS for the Active Storage Class space usage. SSM assumes that all thresholds are "OK" until it receives contrary information.

PVR Cartridge Thresholds

This displays the most severe status of cartridge usage reported by any configured PVR. SSM assumes that all thresholds are "OK" until it receives contrary information.

HPSS Config Changes

This displays whether certain configuration changes have been made that require server(s) to be reinitialized. Clicking the **Apply Config** button when it is enabled will reinitialize the server(s) to apply the configuration changes.

For the **Servers** and **Devices and Drives** fields, possible status values in increasing order of severity are:

- `Normal` indicates no problem reported.
- `Unknown` indicates SSM cannot determine the true status due to communication problems or other difficulties.
- `Suspect` indicates a situation has been encountered by a server or device which might be a problem, but does not significantly affect HPSS operation.
- `Minor` indicates a problem has been encountered by the server or device, but it does not significantly affect HPSS operation.

- `Major` indicates a problem has been encountered by the server or device that may degrade HPSS operation.
- `Critical` indicates a problem has been encountered by the server or device that may disrupt HPSS operation until the problem is resolved.

For the **Storage Class Thresholds** field, the possible status values are:

- `OK` indicates no problem reported.
- `Warning` indicates a threshold has been reported as crossing its warning limit.
- `Critical` indicates a threshold has been reported as crossing its critical limit.
- `Stale` indicates the Migration/Purge Server is down or SSM has not received an update from the MPS.

For the **PVR Cartridge Thresholds** field, the possible status values are:

- `OK` indicates no problem reported.
- `Warning` indicates a threshold has been reported as crossing its warning limit.
- `Critical` indicates a threshold has been reported as crossing its critical limit.
- `Unknown` or `Stale` indicates the PVR is down or SSM has not received an update from the PVR.

For the **HPSS Config Changes** field, the possible status values are:

- `OK` indicates no configuration changes need to be applied.
- `Pending` indicates pending configuration changes need to be applied.
- `Error` indicates an error occurred applying the configuration changes.
- `Unknown` indicates the SSM cannot determine if the configuration changes were applied due to a communication problem or other difficulties.

As problems are resolved or returned to normal, the status fields will automatically reflect the changes.

In addition to the text which describes the status, these fields are displayed with colored icons. The icon color depicts that status as follows:

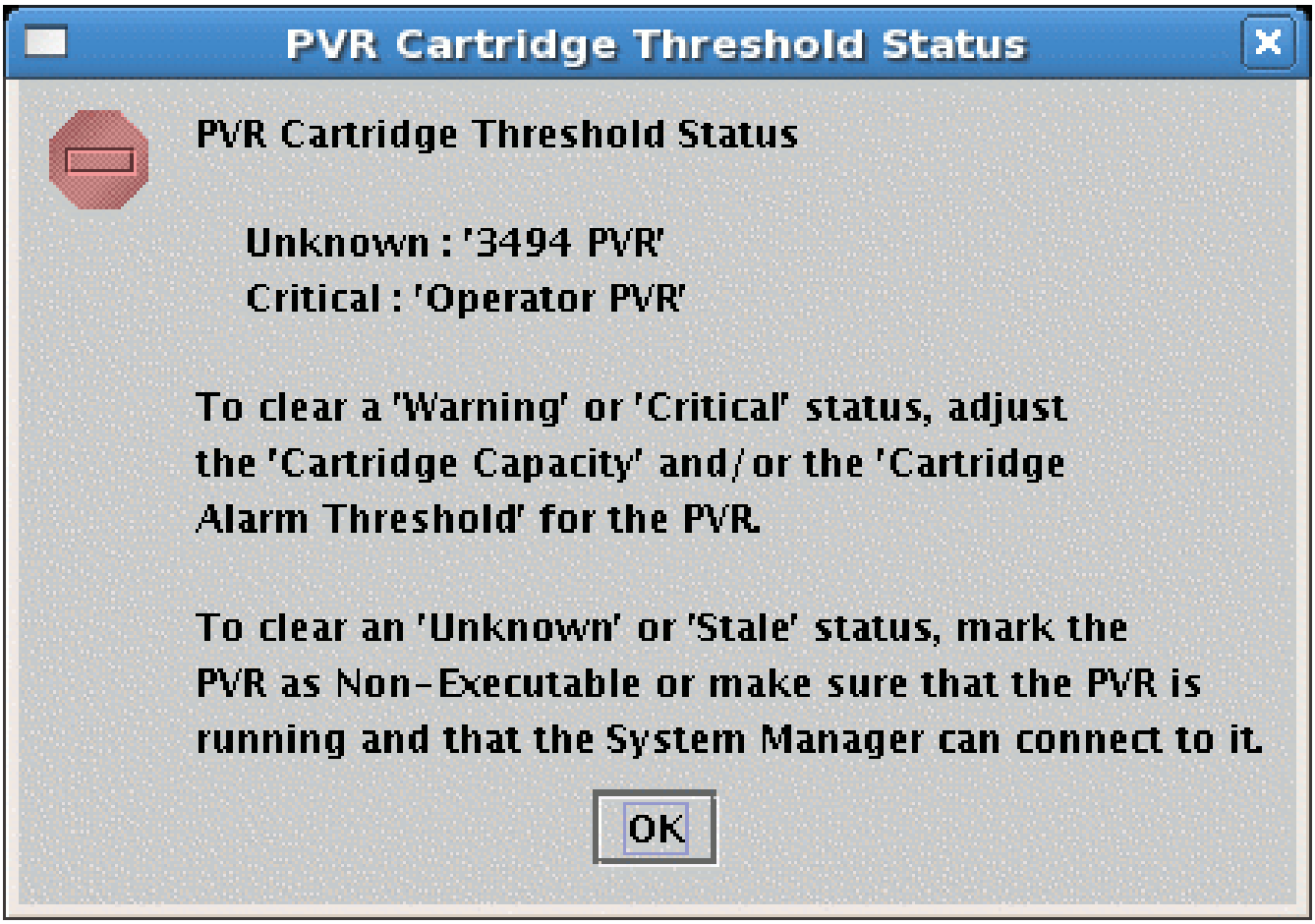
- Red for major and critical problems
- Magenta for minor problems
- Yellow for unknown, stale, suspect, and warning problems
- Green for normal; no problem

Click on the button to the right of the status icon to get more details.

For **Servers**, **Devices and Drives**, and **Storage Class Thresholds** the button will open the corresponding SSM list window in sick list mode. Once this window is open, use it to get detailed status information on the sick entries, assuming the HPSS servers are still healthy enough to respond to SSM requests.

See SSM list preferences for more information on the sick list mode.

For **PVR Cartridge Thresholds** the button will display a message dialog with information about the PVRs that have cartridge threshold issues. This message dialog will look like the following:



The status section of the *HPSS Health and Status* window can be hidden from view by selecting the **View** menu item and unchecking the **HPSS Status** check box.

14.11.3.3. HPSS statistics

The HPSS statistics fields are located in the middle section of the *HPSS Health and Status* window and display the number of bytes moved, bytes used, data transfers, and current PVL jobs in the system. The number of bytes moved and number of data transfers indicate the data accumulated by the Movers since startup or data reset.

HPSS statistics fields show general trends in HPSS operations; the numbers are not all-inclusive. Some values may fluctuate up and down as servers are started or shut down. Some values, such as **Bytes Moved**, can be reset to zero in individual Movers and by SSM users.

Bytes Moved

Total bytes moved as reported by all running Movers.

Bytes Used

Total bytes stored on all disk and tape volumes as reported by all running Core Servers.

Data Transfers

Total data transfers as reported by all running Movers.

PVL Jobs

Total jobs reported by the PVL.

The statistics section of the *HPSS Health and Status* window can be hidden from view by selecting the **View** menu item and unchecking the **HPSS Statistics** check box.

14.11.3.4. Menu tree

The menu tree section of the *HPSS Health and Status* window displays a tree structure that mirrors the structure of the **Monitor**, **Operations**, and **Configure** menus. The menu tree can be fully expanded so that the user can view all the menu options for monitoring, configuring, or operating HPSS. Selecting a leaf of the menu tree results in the same response as selecting the same item from the *HPSS Health and Status* menu bar. The user can also choose to expand only those branches of the menu tree that are of interest. The collapsed and expanded state of the branches on the menu tree can be toggled by clicking on the branch icons.

The menu tree can be hidden from view by selecting the **View** menu item and unchecking the **Menu Tree** check box.

14.11.3.5. File menu

All SSM data windows have the **File** menu. See Common window elements for a description of the **File** menu options that appear on all SSM data windows. The following **File** menu options are unique to the *HPSS Health and Status* window:

Cascade

Select the **Cascade** menu option to rearrange the SSM windows, placing them on the screen starting in the upper left-hand corner and cascading downward toward the lower right-hand corner of the user's desktop. When the cascading has been completed, the *HPSS Health and Status* window will be centered on the screen and brought to the foreground.

Close All

Select **Close All** to close all SSM windows except the *HPSS Health and Status* window. To close the *HPSS Health and Status* window, the user must select **Logoff** or **Exit**.

Logoff

Select **Logoff** to exit the SSM session and close all SSM windows. The hpssgui(7) script will continue to be active and will still use the same user's session log. This will begin the login process again. Note, however, that if the login process used a keytab or existing credentials, this will automatically log the user back in. If this was the case, it is better to choose the Exit option instead.

Exit

Select **Exit** to exit the SSM session and close all SSM windows. The hpssgui(7) script will terminate and the user's session log will be closed. The user must rerun the hpssgui(7) script to access the *HPSS Login* window and reconnect to the System Manager.

14.11.3.6. View menu

The **View** menu is only available on the *HPSS Health and Status* window. The **View** menu offers the user the ability to hide or display elements of the *HPSS Health and Status* window in order to optimize the viewable area. Under the **View** menu there is a menu item and check box for each window element that can be hidden. If the box contains a check mark then the corresponding section of the *HPSS Health and Status* window that displays this element will be visible. If the check box is empty, then the element is hidden from the window view. Clicking on the check box will toggle the visible state of the window element.

HPSS Status

Toggle the **HPSS Status** menu option to hide or view the HPSS status section of the *HPSS Health and Status* window.

HPSS Statistics

Toggle the **HPSS Statistics** menu option to hide or view the HPSS statistics section of the *HPSS Health and Status* window.

Menu Tree

Toggle the **Menu Tree** menu option to hide or view the menu tree section of the *HPSS Health and Status* window.

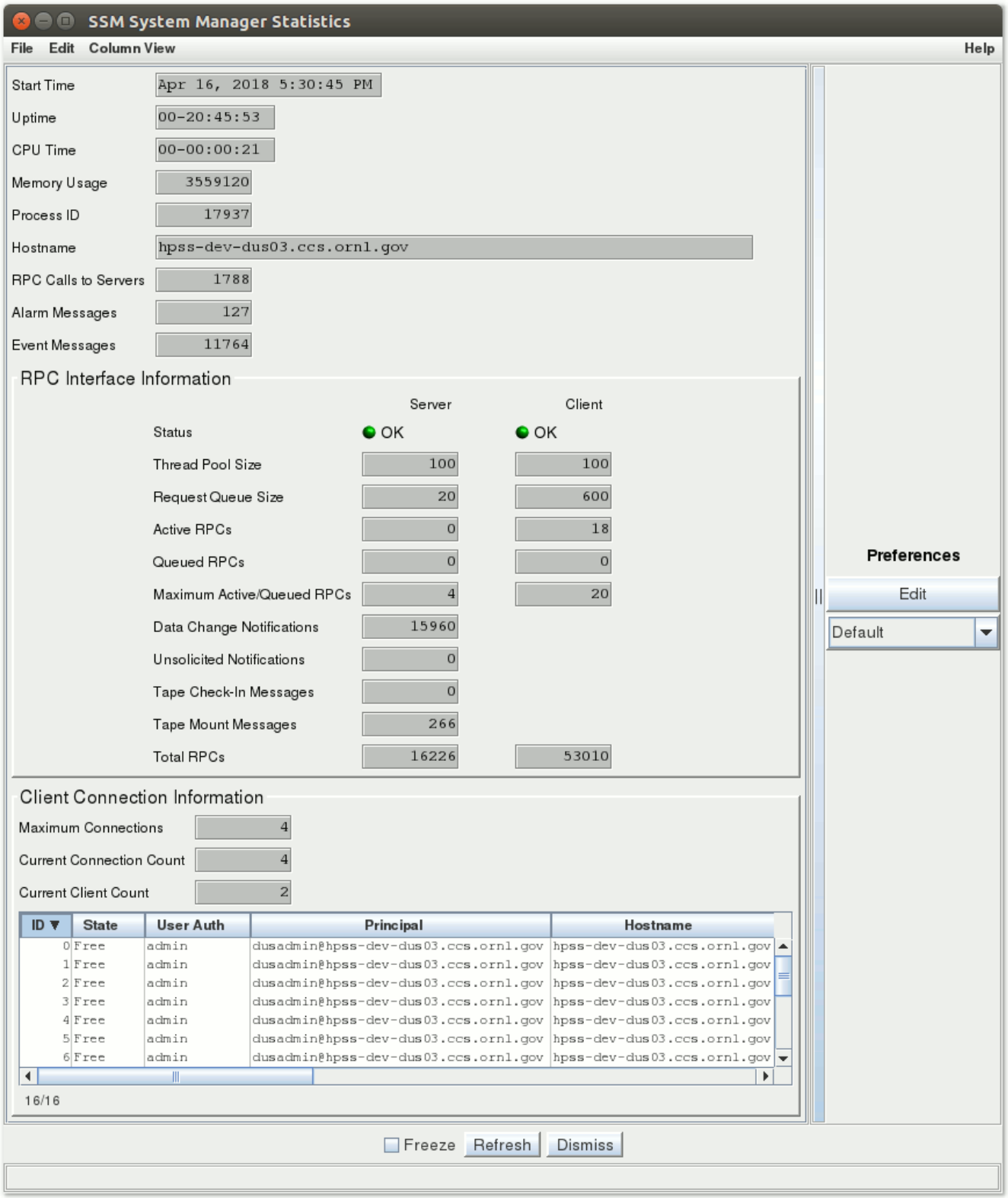
14.11.4. SSM information windows

These windows describe the System Manager and the user's hpssgui(7) session.

14.11.4.1. System Manager Statistics window

This window displays the statistics about the number of RPC calls, notifications, and messages that the System Manager has processed since its Start Time. While the window is open, the statistics are updated at timed intervals. The information in this window is intended for use in analyzing System Manager performance or in troubleshooting. It is of limited use in everyday production situations.

To open the window, start at the *HPSS Health and Status* window. From its menu bar, select **Monitor** > **SSM Information** > **System Manager Statistics**.



Field descriptions

Start Time

The time the System Manager was started.

Uptime

The elapsed clock time since the System Manager was started.

CPU Time

The amount of CPU time that the System Manager has consumed.

Memory Usage

The amount of memory that the System Manager is currently occupying, in kilobytes (KB).

Process ID

The process ID of the System Manager.

Hostname

The name of the host where the System Manager is running.

RPC Calls to Servers

The number of RPCs the System Manager has made to other HPSS servers.

Alarm Messages

The number of alarm-type messages read from the HPSS syslog file.

Event Messages

The number of event-type messages read from the HPSS syslog file.

RPC Interface Information

Provides information about the server and client RPC interfaces. The server interface is used by other HPSS servers to contact the System Manager. The client interface is used by the SSM client to contact the System Manager. There are two columns of data, one for the server interface and one for the client interface. Not all fields are available for both interfaces. The fields include:

Status

Current status of the thread pool and request queue of the RPC interface. The Status can be:

- `OK` indicates the number of active RPCs is less than the **Thread Pool Size**. There are enough threads in the thread pool to handle all current RPCs with spare ones left over.
- `Warning` indicates the number of active RPCs is greater than the **Thread Pool Size**. The number of queued RPCs is less than 90% of the **Request Queue Size**. There aren't enough threads to handle all the current RPCs and some are having to wait in the queue, but the queue is big enough to hold all the waiters.
- `Critical` indicates the number of queued RPCs is greater than or equal to 90% of the **Request Queue Size**. There aren't enough threads to handle all the current RPCs, some are having to wait in the queue, and the queue is getting dangerously close to overflowing, at which point any new RPCs will be rejected.

Thread Pool Size

The maximum number of RPCs that can be active at any one time. For the server RPC interface this value is determined by the HPSS_SM_SRV_TPOOL_SIZE environment variable. For the client RPC interface this value is determined by the **Thread Pool Size** field defined on the *Core Server Configuration* window. Refer to Interface controls.

Request Queue Size

The maximum number of RPC requests that can be queued and waiting to become active. For the server RPC interface this value is determined by the HPSS_SM_SRV_QUEUE_SIZE environment variable. For the client RPC interface this value is determined by the **Request Queue Size** field on the *Core Server Configuration* window. Refer to Interface controls.

Active RPCs

The number of RPCs that are currently active. To be active an RPC must have been assigned to a thread in the thread pool.

Queued RPCs

The number of RPCs that are waiting in the request queue to be assigned to a thread in the thread pool.

Maximum Active/Queued RPCs

The maximum number of RPC requests that were active (in the thread pool) or queued (in the request queue) at the same time. This value can be used to help tune the **Thread Pool Size** and **Request Queue Size** for the RPC interface. If the **Maximum Active/Queued RPCs** is greater than the **Thread Pool Size** you might consider increasing the **Thread Pool Size**, the **Request Queue Size**, or both, to help with the System Manager performance. However, increasing these two parameters could cause the System Manager to require more memory.

Data Change Notifications

The number of data change notifications received from servers. (Server RPC Interface only).

Unsolicited Notifications

The number of notifications which the System Manager received from other HPSS servers but which it did not request from them. (Server RPC Interface only).

Tape Check-In Messages

The number of tape check-in request notifications received. (Server RPC Interface only).

Tape Mount Messages

The number of tape mount request notifications received. (Server RPC Interface only).

Total RPCs

Total number of RPCs processed by the RPC interface.

Client Connection Information

Provides information about clients that have connected to the System Manager.

Maximum Connections

The maximum number of client connections that the System Manager was handling at any one time. Each client can have multiple connections. The default connections per client is "2". Each client can specify the number of connections using the -Dhpss.ssm.SMConnections Java command line option.

Current Connection Count

The number of client connections currently being handled by the System Manager.

Current Client Count

The number of clients currently connected to the System Manager. This will be the number of entries in the **Client List** (below) with an `In Use` state.

Client List

The list of clients connected to the System Manager. The entries that appear in the client list will be for (up to) the last 64 (SSM_CLIENT_MAX) clients that have connected to the System Manager.

ID

Slot identifier.

State

Clients displayed in the client list can have one of two states: `In Use` and `Free`.

- `In Use` indicates the clients that are currently connected to the System Manager.
- `Free` indicates the clients that were once connected but are no longer connected (active).

  Once the client list contains SSM_CLIENT_MAX `In Use` and/or `Free` entries, then the oldest `Free` slot (the client that has been disconnected the longest) is given to the next client. Once all the slots are `In Use`, then the client table is full; no new clients can connect until one of the `In Use` slots becomes `Free` after a client disconnects.

User Auth

The client user's authorization level: `admin` or `operator`. See Configuration and startup of **hpssgui** and **hpssadm** for more details.

Principal

The user's principal name.

Hostname

The name of the host where the client is running.

Connections

The number of RPC connections this client has to the System Manager.

Start Time

The time that the client connected to the System Manager.

Connect Time

The elapsed time since the client connected to the System Manager.

Idle Time

The elapsed time since the System Manager received an RPC from the client.

Cred Refreshes

The number of times the principal's credentials have been refreshed since the client has been connected to the System Manager.

RPC Calls

The number of RPCs that the client has made to the System Manager.

RPC Waits

The number of RPCs that the client has currently "waiting" in the System Manager. These are the RPCs which are active and connected to the System Manager but which have not yet completed.

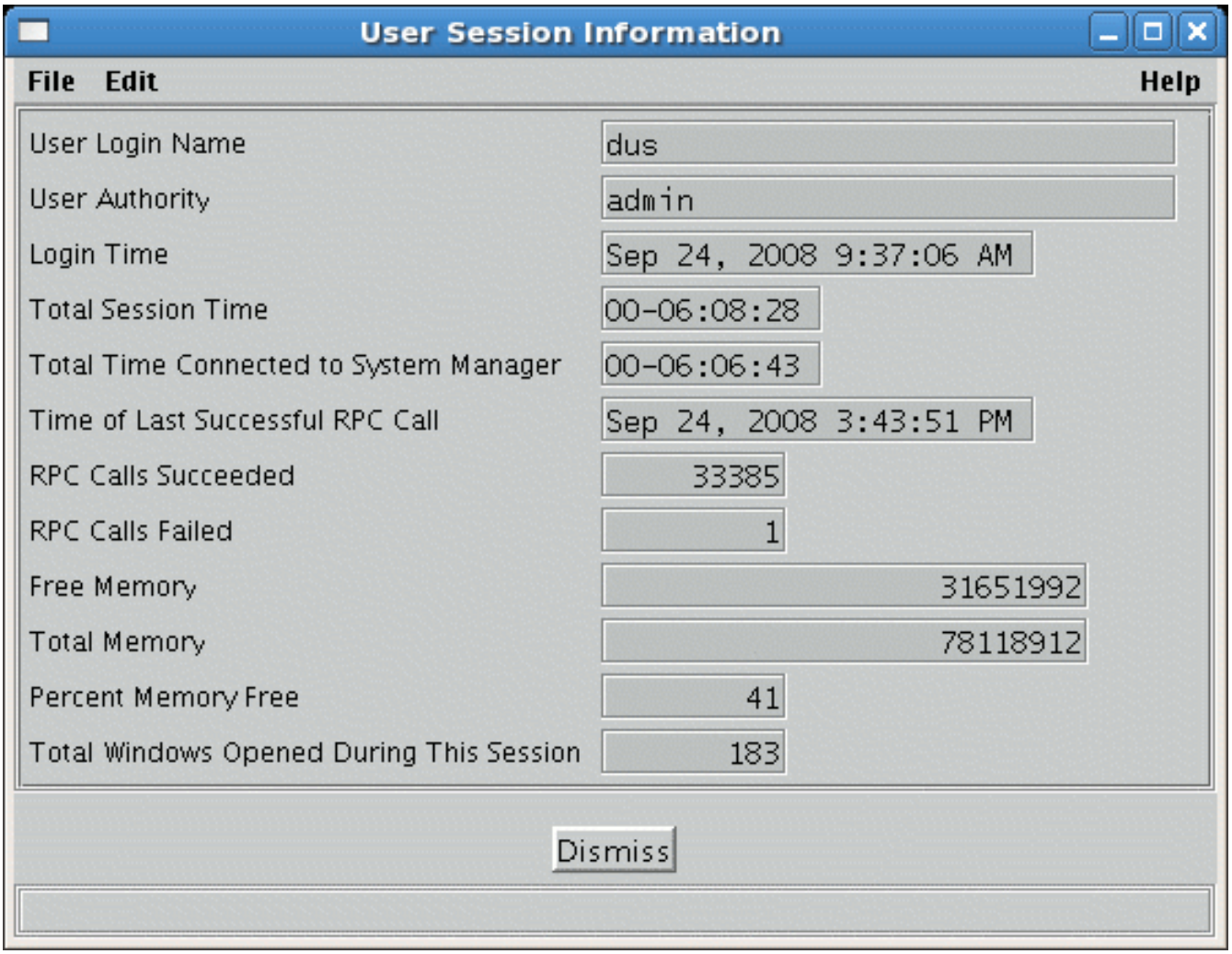
Client UUID

The UUID for the client.

14.11.4.2. User Session Information window

The *User Session Information* window displays memory and network statistics for the current SSM user session. The window is updated in timed intervals as data changes.

To open the window, start at the *HPSS Health and Status* window. From its menu, select **Monitor** > **SSM Information** > **User Session Information**.



Field descriptions

User Login Name

The name that the user entered as the user ID when logging into HPSS.

User Authority

The authority level that the user has in order to perform operations in SSM. This can be admin or operator.

Login Time

The time that the user logged into SSM.

Total Session Time

The amount of time that the user has had the graphical user interface running.

Total Time Connected to System Manager

The amount of time that the user has been connected to the System Manager. This is the time that has elapsed since the login was successful.

Time of Last Successful RPC Call

The time that the last RPC call was made to the System Manager which completed successfully.

RPC Calls Succeeded

The number of successful RPC calls made by this user session.

RPC Calls Failed

The number of unsuccessful RPC calls made by this user session.

Free Memory

Amount of free memory available in the hpssgui(7) program.

Total Memory

Amount of total memory available in the hpssgui(7) program.

Percent Memory Free

The ratio of free memory to total memory in the hpssgui(7) process.

Total Windows Opened During This Session

The number of windows created during the current user session.

14.12. SSM list preferences

When a user logs into SSM, the System Manager reads the saved preferences file and loads them into the SSM session, if they exist.

Each SSM list type has a Default preferences record. The Default preferences configuration is set so that the more commonly used columns are displayed. The following lists have a preferences record:

- Alarms & Events
- Classes of Service
- Devices & Drives
- File Families
- Filesets & Junctions
- Hierarchies
- Logging Policies
- Migration Policies
- Purge Policies
- PVL Jobs
- Object Store Accounts
- Object Store Account Identities
- Object Store Account Buckets
- Object Store Account Mappings
- RTM Summary
- Servers
- SM Clients
- Storage Classes, Active
- Storage Classes, Configured
- Subsystems
- Tape Check-Ins
- Tape Drive Quotas
- Tape Mounts

The *Servers*, *Devices and Drives*, and *Active Storage Classes* list windows also have Sick preferences which display all items with abnormal status.

Default preferences can be modified but cannot be deleted. Sick preferences may not be modified or deleted.

Preferences are saved on the client node in the directory specified by the "configuration path" hpssgui(7) command line argument ("-i"). This option can also be set using the environment variable HPSSGUI_USER_CFG_PATH or the configuration file entry HPSS_SSM_USER_PREF_PATH. If this option is not specified, the default value is `<client node>:<user.home>/hpss-ssm-prefs`. The user must have permissions to create the preferences file in the directory.

Preferences windows contain filters for controlling the data displayed in the list window. Columns in the list window can be rearranged and resized by dragging columns and column borders on the list window itself. Data in the list can be sorted ascending or descending based on any column in the list by clicking in that column's header. Such changes are kept up to date in the preferences file,

without the need to click on a **Save** button.

Columns can be hidden or redisplayed through the **Column View** menu on each list window.

The *Tape Mount Requests* and *Tape Check-In Requests* windows have **Auto Popup** check boxes. The states of these check boxes, as well as those in the **View** Menu of the *HPSS Health and Status* window are stored in the preferences file.

If a user's preferences file becomes obsolete, the current version of the user interface software will convert the old preferences to the current format.

Check box filters

Check box filters apply to columns that have a limited set of display values. The check box filters are grouped by column name and contain a check box for each allowed value. If the check box is selected, all rows containing the value will be displayed on the corresponding SSM list window. At least one check box filter must be selected in each group.

Text filters

Columns that can be filtered by their text content are listed in the Text Filters section of the preference window. Users can control which rows are displayed by entering a Java regular expression into one or more of the **Text Filter** fields. If the **Text Filter** field is blank then no filtering on the field will occur.

The preference window will verify that the text entered is a valid Java regular expression before allowing the user to save the preferences.

For example, if the list contains storage class names as follows

```
SClass_4-way_Disk
SClass_4-way_Tape
Sclass_4-way_Archive
Sclass_Disk
```

entering a Java regular expression of ".**4-way.**" would result in the "Sclass_Disk" entry to be filtered out of the display.

For a complete description on how to use regular expressions in Java, Refer to the following web page: http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html.

Button options

Save as

Save the current preference settings to a new named preference. A pop-up window will be displayed so that the new name can be entered.

Save

Save the current preference settings to the preferences file using the same preference name.

Delete

Delete the currently displayed preference settings from the preferences file. Sick and Default preference settings cannot be deleted.

Reload

Reread the preference settings from the preferences file.

Apply

Apply the current preference configuration to the parent SSM list window. Clicking **Apply** does not save the changes to the preferences file. To save the preference settings, click the **Save** or **Save as** button. The **Apply** button is only available when the Preferences window is accessed via its parent SSM List window (that is, it is not available when the Preferences window is accessed from the **Configure** menu).

Show List

Brings the window to which these preferences apply to the foreground. The **Show List** button is only available when the Preferences window is accessed via its parent SSM List window.

Dismiss

Close the window.

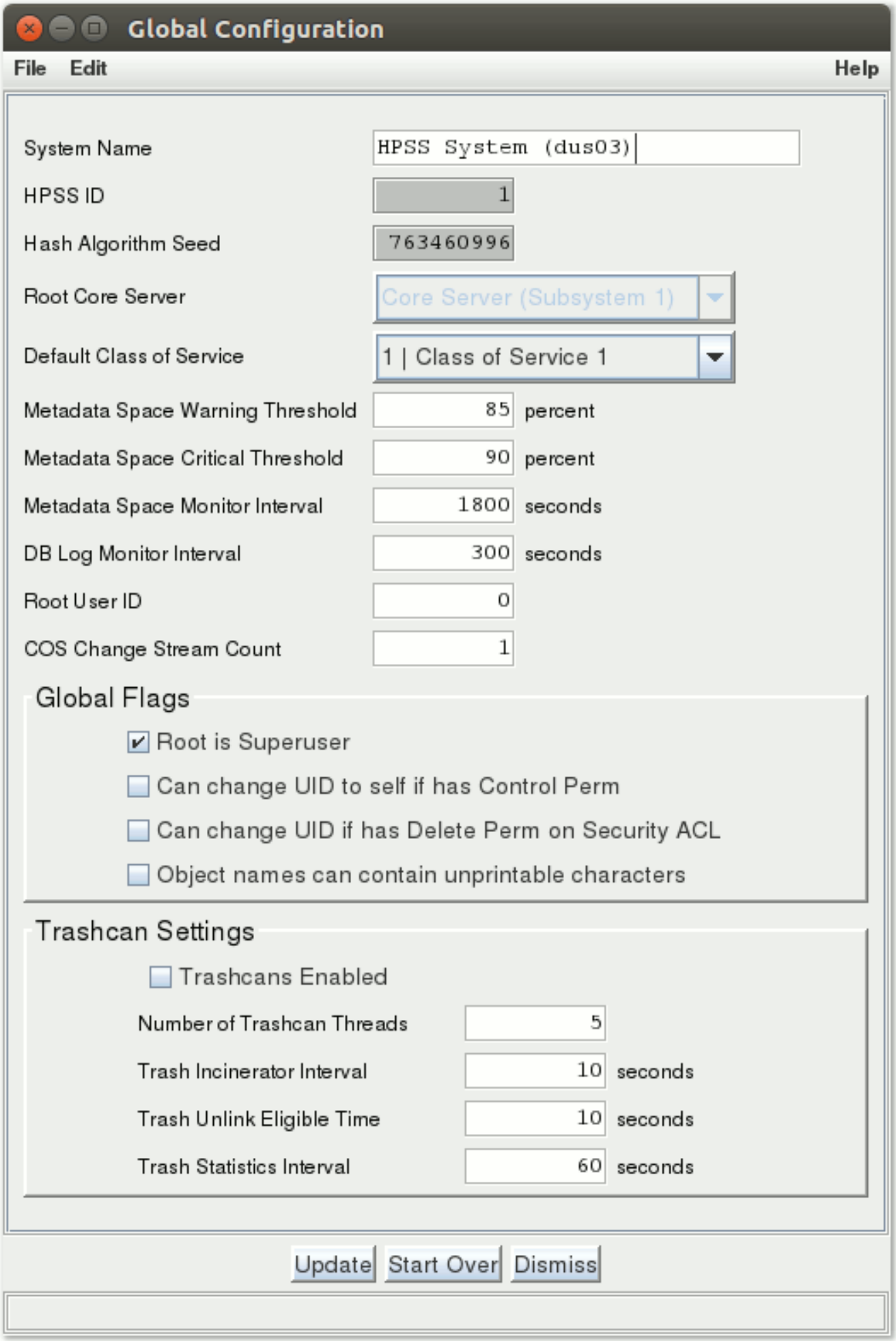
To modify an existing preference configuration, select the Preference Name from the drop down list and click the **Reload** button. Modify the configuration then click the **Save** button. To create a new Preference configuration, open an existing configuration, make the modifications, click the **Save as** button then enter a unique value for the new preference name and click the **OK** button.

Chapter 15. Global and subsystem configuration

This chapter discusses two levels of system configuration: global and storage subsystem. The global configuration applies to the entire HPSS installation while subsystem configurations apply only to servers and resources allocated to storage subsystems.

For new HPSS systems, it is recommended that the first step of configuration be the partial definition of subsystems. The subsystems should be defined at this point except for their Gatekeeper and default COS information. The global configuration should be created after server configuration is complete. Last of all, the subsystem configuration is completed by adding the definitions of the Gatekeeper and default COS. See HPSS configuration roadmap (new HPSS sites).

15.1. Global Configuration window


Global Configuration
File
Edit
Help
System Name
HPSS System (dus03)
HPSS ID
1
Hash Algorithm Seed
763460996
Root Core Server
Core Server (Subsystem 1)
Default Class of Service
1 | Class of Service 1
Metadata Space Warning Threshold
85
percent
Metadata Space Critical Threshold
90
percent
Metadata Space Monitor Interval
1800
seconds
DB Log Monitor Interval
300
seconds
Root User ID
0
COS Change Stream Count
1
Global Flags
Root is Superuser
Can change UID to self if has Control Perm
Can change UID if has Delete Perm on Security ACL
Object names can contain unprintable characters
Trashcan Settings
Trashcans Enabled
Number of Trashcan Threads
5
Trash Incinerator Interval
10
seconds
Trash Unlink Eligible Time
10
seconds
Trash Statistics Interval
60
seconds
Update
Start Over
Dismiss

This window allows you to manage the HPSS global configuration record. Only one such record is permitted per HPSS installation. To open the window, on the *HPSS Health and Status* window select the **Configure** menu, and from there the **Global** menu item.

Field descriptions

System Name

An ASCII text string representing the name for this HPSS system.

HPSS ID

The unique ID assigned to each HPSS installation.

Hash Algorithm Seed

The seed value used to define the system's partition hashing algorithm. The partition hashing algorithm is used to determine the database partition in which the metadata for each object (such as, name space object, bitfile, and storage segment) is stored.



Once defined, the **Hash Algorithm Seed** must never be changed.

Root Core Server

The name of the Core Server which manages the root fileset ("/") in the HPSS name space.



Advice

The root Core Server should be selected with care. Once it is chosen it cannot be changed as long as the chosen server exists. If the root Core Server must be changed, the current root Core Server will have to be deleted before SSM will allow another root Core Server to be selected.



The following settings are global defaults. If they are changed, the Core Servers which do not override these defaults will all need to be reinitialized or restarted in order to actualize the new settings. It is recommended to use the **Apply Config** button on the *HPSS Health and Status* window to reinitialize the server(s).

Default Class of Service

The name of the default Class of Service (COS). Core Servers will store new files in this COS when the HPSS client does not specify a COS or provide hints on the creation request. This default can be overridden by each storage subsystem.



Advice

If the COS chosen for the default Class of Service is deleted, be sure to change the default Class of Service before deleting the COS.

If disk is used as the default COS, it should be defined so that a large range of file sizes can be handled. For this reason it may be reasonable to set the default COS to a tape COS.

Metadata Space Warning Threshold

Provides a default value for the metadata warning threshold. When the space used in any DB2 tablespace exceeds this percentage, the Core Server will issue a periodic warning message. This value can be overridden for each storage class on a storage subsystem basis (see Subsystems list window for more information).

Metadata Space Critical Threshold

Provides a default value for the metadata critical threshold. When the space used in any DB2 tablespace exceeds this percentage, the Core Server will issue a periodic critical alarm. This value can be overridden for each storage class on a storage subsystem basis (see Subsystems list window for more information).

Metadata Space Monitor Interval

The Core Server will check metadata usage statistics at the indicated interval, specified in seconds. The minimum value for this field is 1800 seconds (30 minutes). A value of zero will turn off metadata space monitoring. This field may be overridden on the Storage Subsystem configuration.

DB Log Monitor Interval

The Core Server will check the consistency of Database Logs and Backup Logs at the indicated interval, specified in seconds. The logs are consistent if both primary and backup log directories exist and contain log files with the same names. The minimum value for this field is 300 seconds (five minutes). A value of zero will turn off DB Log monitoring. This field may be overridden on the Storage Subsystem configuration.

Root User ID

The UID of the user who has root access privileges to the HPSS name space. This only applies if the **Root Is Superuser** flag is set.

COS Change Stream Count

The number of background threads that run in the Core Server to process Class of Service change requests. This field may be overridden on the Storage Subsystem configuration.

Global flags

Root Is Superuser

If checked, root privileges are enabled for the UID specified in the **Root User ID** field. If the box is not checked, the UID specified in the **Root User ID** field will not have root privileges. Root access privileges grant the specified user the same access rights to each name space object as the object's owner.

Can change UID to self if has Control Perm

If checked, then when a user has control access to an object, the user can take over ownership of the object by changing the UID of the object to the UID associated with the user.

Can change UID if has Delete Perm on Security ACL

If checked, then when a user is listed with delete permission in the security ACL of the Core Server that owns the object, the user can change the UID of the object to any valid UID.

Object names can contain unprintable characters

If checked, then users of the HPSS system may create files, directories, or other objects with names containing unprintable characters as viewed through the 7-bit ASCII character set. If this option is off, any attempt to name an object using unprintable characters will be disallowed. The range of printable characters is 0x20 (blank) through 0x7E (tilde) in the 7-bit ASCII character set.

Trashcan settings

Trashcans Enabled

If checked, the HPSS trashcan feature will be enabled after the Core Servers are recycled or reinitialized.

Number of Trashcan Threads

Specifies the number of background threads that are to be employed deleting entries from trashcans. This is a performance-tuning knob. For example, if the number of entries in trashcans seems to be high, or if there is a need to reclaim disk space, or if there is some other trashcan-related need, the administrator may wish to increase the number of trashcan-related background threads so that these tasks can be performed more quickly. On the other hand, if the number of trashcan background threads is taking up resources that might be needed elsewhere, the administrator may wish to decrease the number of threads.

Trash Incinerator Interval

The trashcan deletion threads run periodically in the background. This knob is provided to control the periodicity. **Trash Incinerator Interval** is specified in seconds. For example, if this setting is 3600 the trashcan deletion threads will kick off every hour.

Trash Unlink Eligible Time

The amount of time, in seconds, that an entry will remain in the trashcan before it is eligible for permanent deletion by the garbage collector. For example, if **Trash Unlink Eligible Time** is set to 691200 then entries would be allowed to remain in trashcans for up to 8 days. The default setting is 864000 (10 days).

Trash Statistics Interval

The thread that gathers trashcan statistics runs periodically to gather statistics about all entries that are currently in trashcans. This knob controls how often this thread runs and is specified in seconds. For example, if **Trash Statistics Interval** is set to 7200 the trashcan statistics thread would run every two hours.



Advice

*The administrator should be aware that the trashcan statistics thread could consume significant system resources when run on a system having a large number (millions) of entries in trashcans. In such a case the administrator might wish to increase the **Trash Statistics Interval** value so that the thread runs less often.*

*When changes are made to trashcan settings, the Core Servers must be reinitialized or restarted for the new settings to take effect. It is recommended to use the **Apply Config** button on the*

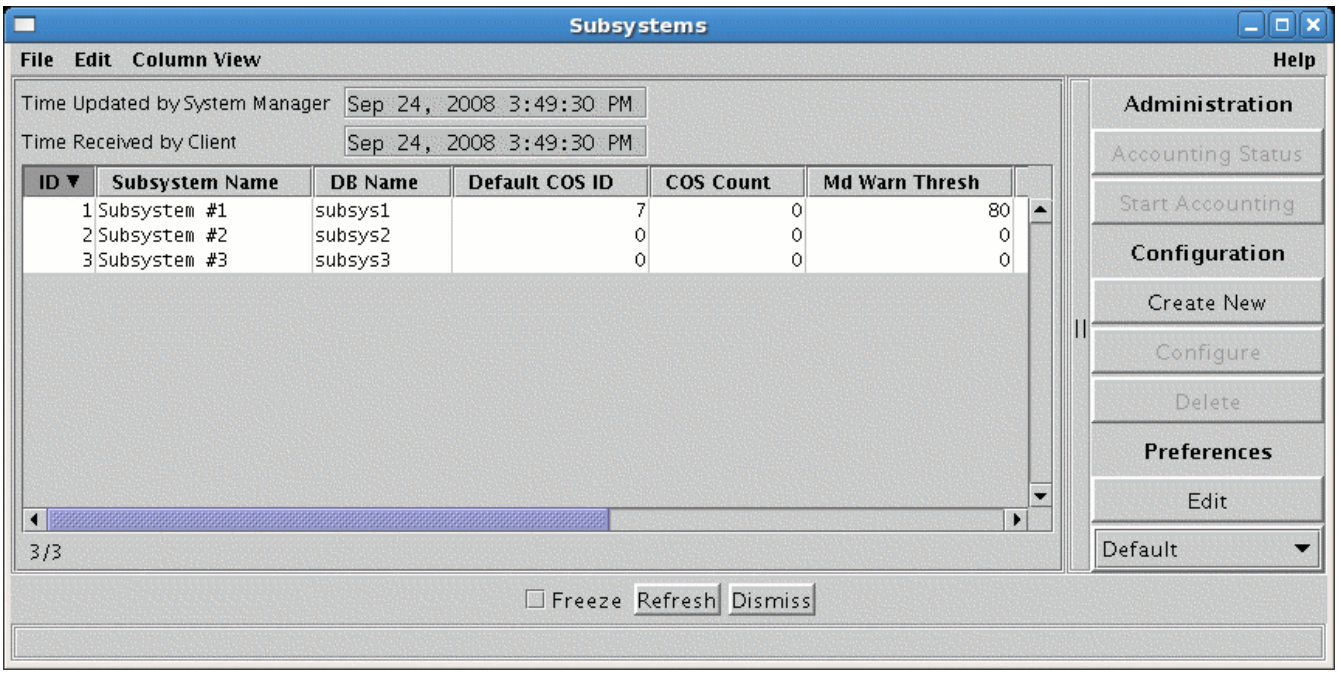
HPSS Health and Status window to reinitialize the server(s).

15.2. Storage subsystems

Every HPSS system consists of at least one subsystem. This chapter provides instructions and supporting information for creating, modifying, and deleting them.

For a conceptual overview of subsystems, refer to the sections Storage subsystems and HPSS storage subsystems sections.

15.2.1. Subsystems list window



This window lists all the subsystems in the HPSS system and provides the ability to manage these subsystems. To open the window, from the *Health and Status* window select the **Configure** menu, and from there select the **Subsystems** menu item.

To create a new subsystem, click on the **Create New** button. To configure an existing subsystem, select it from the list and click on the **Configure** button. When creating or configuring a subsystem, the *Storage Subsystem Configuration* window will appear.

To delete an existing subsystem, select it from the list and click on the **Delete** button.

Field descriptions

Subsystem List table

For details on the columns listed for each subsystem, refer to Storage Subsystem Configuration window.

Administration buttons

Accounting Status

Opens the *Accounting Status* window which displays the status and statistics from the last

accounting run. See Generating an accounting report for more information. You must first select a configured subsystem by clicking on the subsystem entry from the list of displayed subsystems.

Start Accounting

Start the Accounting utility to generate an accounting report. See the *Accounting Status* window for updates from the utility. You must first select a subsystem by clicking on the subsystem entry from the list of displayed subsystems.

Configuration buttons

Create New

Create a new storage subsystem definition by opening a *Storage Subsystem* window containing default values for a new subsystem.

Configure

Open the selected subsystem configuration for editing. This button is disabled unless a subsystem is selected in the list.

Delete

Delete the selected subsystems. This button is disabled unless a subsystem is selected in the list.



Always contact HPSS support before deleting a subsystem definition. An improperly deleted subsystem can cause serious problems for an HPSS system. Refer to Deleting a storage subsystem.

Related information

Storage subsystems

HPSS storage subsystems

Generating an accounting report

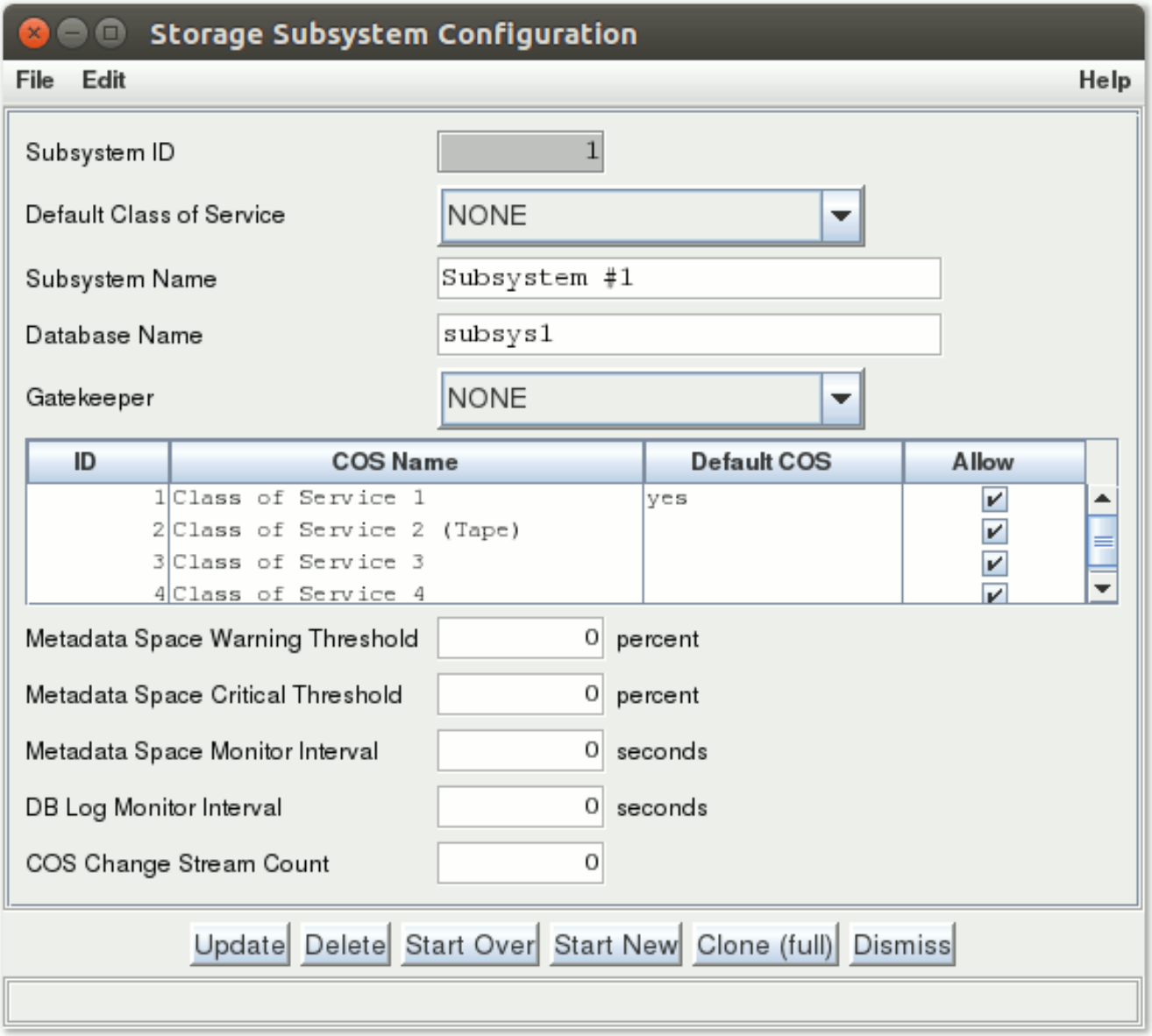
15.2.2. Creating a new storage subsystem

The steps detailed in the following sections should be followed to create a new storage subsystem within an HPSS system. These steps are:

1. Create the storage subsystem metadata Create storage subsystem metadata
2. Create the storage subsystem configuration Create storage subsystem configuration
3. Create the storage subsystem servers Create storage subsystem servers
4. Assign a Gatekeeper to the subsystem, if required Assign a Gatekeeper if required
5. Assign storage resources to the storage subsystem Assign storage resources to the storage subsystem
6. Create the storage subsystem fileset and junction Create storage subsystem fileset and junction
7. Configure migration and purge policy overrides Migration and purge policy overrides

8. Configure storage class threshold overrides Storage class threshold overrides

15.2.3. Storage Subsystem Configuration window



This window allows an administrator to manage the configuration of a storage subsystem.

The **Add** button is only displayed during the creation of a new configuration. The **Update** button is displayed when an existing configuration is being modified.

To open this window for creation of a new subsystem, click the **Create New** button on the *Subsystems* window. To open this window for an existing subsystem, select the subsystem from the *Subsystems* window and click the **Configure** button.

Field descriptions

The following fields may be set upon initial creation of the subsystem or modified as configuration needs change.



After making any changes to the following fields, the Core Server for this subsystem

*will need to be reinitialized or restarted to load the new configuration. It is recommended to use the **Apply Config** button on the HPSS Health and Status window to reinitialize the server(s).*

Subsystem ID

A unique, positive integer ID for the storage subsystem. This field may only be set at create time. The default value is the last configured subsystem ID number plus "1". The default subsystem ID can be overwritten but if the new number already exists, an attempt to save the configuration will fail.

Default Class of Service

This value overrides the global Default Class of Service specified on the *Global Configuration* window. (Refer to Global Configuration window)

This default Class of Service applies to this storage subsystem only. The default value is None.



Advice

HPSS Core Servers use a default Class of Service to store newly-created files when the HPSS user does not specify a COS or any hints with the creation request. The global configuration specifies a default COS for an entire HPSS installation. Selecting a COS on the Storage Subsystem Configuration window allows the global value to be overridden for a particular subsystem.

If the field is blank, the global default COS will be used. If no Classes of Service are configured, this value can be updated after the Classes of Service are in place.

Subsystem Name

The descriptive name of the storage subsystem. This field may be set only when the storage subsystem is created. The name should be unique and informative. It can contain a character string up to 31 bytes long. The default value is "Subsystem #<ID>".

Database Name

The name of the database to be used to store the metadata for the storage subsystem.

Gatekeeper

The default value is None.



Advice

If an appropriate Gatekeeper has not yet been configured, simply leave this configuration entry blank. It can be updated after the Gatekeeper is in place.

Allowed COS list

A list of Classes of Service that can be used by this subsystem. To allow a COS to be used by this subsystem, the corresponding check box must be selected in the Allow column of the list. At least one COS must always be selected. The user will not be permitted to deselect the COS defined to be the Default Class of Service. If this subsystem configuration does not have a Default Class of Service defined, then the COS chosen as the global configuration's Default Class of Service

cannot be deselected.



*Note that a newly created COS will not appear in the selection list until the changes have been applied by clicking the **Apply Config** button on the HPSS Health and Status window or the Core Server and Migration/Purge Server associated with the subsystem have been reinitialized or recycled. When new Classes of Service are added, the initial allowed state for that COS is determined by the current setting for the other Classes of Service. If all previous Classes of Service were allowed, the new COS will be allowed; otherwise, the new COS will be disallowed.*

Advice

By default, the servers in a subsystem are able to use any configured COS. This table allows an administrator to prevent a subsystem from using particular Classes of Service.



When a new Class of Service is added to a system, it will automatically be enabled for all subsystems which have no disabled Classes of Service. It will be disabled in all other subsystems. If this is not the desired configuration, the COS will have to be allowed or disallowed for each subsystem individually.

Disallowing all Classes of Service in a subsystem is not permitted.

Metadata Space Warning Threshold

The Core Server in this subsystem will issue a warning alarm periodically and set its Opstate to Major when the percentage of used space in any DB2 tablespace in this subsystem's database exceeds this value. If a value of "0" is specified, the global configuration setting will be used.

Metadata Space Critical Threshold

The Core Server in this subsystem will issue a critical alarm periodically and set its Opstate to Critical when the percentage of used space in any DB2 tablespace in this subsystem's database exceeds this value. If a value of "0" is specified, the global configuration setting will be used.

Metadata Space Monitor Interval

The Core Server for this subsystem will check the metadata usage statistics at the indicated interval, specified in seconds. If a value of "0" is specified, the global configuration setting will be used for this storage subsystem. The minimum value for this field is 1800 seconds (30 minutes).

DB Log Monitor Interval

The Core Server will check the consistency of Database Logs and Backup Logs at the indicated interval, specified in seconds. The logs are consistent if both primary and backup log directories exist and contain log files with the same names. The minimum value for this field is 300 seconds (five minutes). If a value of "0" is specified, the global configuration setting will be used for this storage subsystem.

COS Change Stream Count

The number of background threads that run in the Core Server to process Class of Service change requests. If a value of "0" is specified, the global configuration setting will be used for this

storage subsystem.

15.2.3.1. Create storage subsystem metadata

Before creating the subsystem metadata, you must have created the subsystem database that will be used by the subsystem and have created appropriate tablespaces for the database.

You should review and perform the steps in the following sections before allocating metadata space for the new subsystem:

- Configuring HPSS storage subsystems
- Setting up HPSS infrastructure storage space
- Information on HPSS and DB2 file systems
- Information on HPSS metadata space
- How to Configure HPSS - secondary subsystem machine

15.2.3.2. Create storage subsystem configuration

Use the following steps to create a storage subsystem configuration:

1. Decide on a unique descriptive name for the new storage subsystem. SSM will automatically choose the name "Subsystem #N", where N is the subsystem ID selected by SSM. The name of the new storage subsystem may be changed by the administrator at subsystem configuration time only.
2. From the *HPSS Health and Status* window, select **Configure** from the menu. Then select **Subsystems**. You will then be presented a window that lists currently configured subsystems. Select the **Create New** button to bring up another window to configure the new subsystem.
3. Enter the name you have chosen for the subsystem in the **Subsystem Name** field and enter the ID of the subsystem in the **Subsystem ID** field. Enter the name of the database you have chosen to contain the tables for this subsystem in the **Database Name** field.
4. Decide which Classes of Service are to be supported by the new storage subsystem. SSM will automatically select all Classes of Service to be supported, but the administrator can modify these choices at any time. Also, decide on a default Class of Service for this storage subsystem. By default SSM leaves this field blank, which means that the default Class of Service specified in the global configuration will apply to the new storage subsystem as well. The administrator may choose to override the global value by using the subsystem configuration value at any time.
5. Decide whether gatekeeping or account validation are needed for this storage subsystem. If either is required, a Gatekeeper will need to be configured for this storage subsystem. If the required Gatekeeper is already configured, simply add it to your storage subsystem's configuration. However, if it is not yet configured, it will be necessary to wait until Assign a Gatekeeper if required to add the Gatekeeper.
6. Set the metadata space thresholds and the update interval. Typical values are "75" for warning, "90" for critical and "300" to have the metadata space usage checked every 300 seconds.
7. Set the **DB Log Monitor Interval**. The minimum value is 300 seconds; the typical value is 1800 seconds.

8. Click the **Add** button to store the configuration.

15.2.3.3. Create storage subsystem servers

The new storage subsystem must contain a single Core Server. If migration and purge services will be needed then a Migration/Purge Server is also required. See the following sections for instructions on configuring these new servers:

Server configuration

Core Server-specific configuration

Migration/Purge Server (MPS)-specific configuration

On each server's basic configuration window, be sure to assign the server to the new storage subsystem.

15.2.3.4. Assign a Gatekeeper if required

If gatekeeping or account validation are needed and an appropriate Gatekeeper is not already configured, a new Gatekeeper will be required. See the Gatekeeper section and Gatekeeper-specific configuration for instructions on configuring the Gatekeeper.

Be sure to assign this Gatekeeper to the appropriate storage subsystem by choosing it from the Gatekeeper selection list on the appropriate Storage Subsystem Configuration window.

15.2.3.5. Assign storage resources to the storage subsystem

Storage resources are tied to a storage subsystem through storage classes. To determine which storage classes belong to a particular subsystem, look up the configuration for each Class of Service available to the storage subsystem. Each Class of Service is tied to one storage hierarchy which is in turn tied to some number of storage classes. Storage resources must then be assigned to the storage classes which are used in the referenced hierarchies. See Creating storage resources for details.

15.2.3.6. Create storage subsystem fileset and junction

The HPSS name space consists of filesets that are connected by junctions. The top (root directory) of the HPSS name space is the RootOfRoots fileset managed by the root Core Server. The rest of the name space is built by pasting filesets together using junctions.

Since each subsystem has its own Core Server, it also has its own root fileset. This fileset needs to be connected to the HPSS name space in order for files to be stored in the new subsystem. This is accomplished by creating a junction from the HPSS name space to the root fileset of the new subsystem.

See the Filesets and junctions chapter for more information.

15.2.3.7. Migration and purge policy overrides

The migration and purge policies contain two elements, the basic policy and the storage subsystem-specific policies. This can be seen on the *Migration Policy* and *Purge Policy* windows. If a given

migration or purge policy does not contain any subsystem-specific policies, then the basic policy applies across all storage subsystems and no other configuration is needed. If it is desired for migration or purge to behave differently than specified in the basic policy in a given storage subsystem, then a storage subsystem-specific policy should be created for that subsystem. A subsystem-specific policy allows some or all of the values in the basic policy to be overridden for the given storage subsystem. See Migration policy configuration and Purge Policy configuration for more information.

15.2.3.8. Storage class threshold overrides

The warning and critical thresholds given on the *Storage Class Configuration* window apply across all storage subsystems unless specified otherwise. The **Subsystem Thresholds** button on the *Configured Storage Class* list window allows the default thresholds to be overridden for specified storage subsystems. See Configured Storage Classes window for more information.

15.2.4. Modifying a storage subsystem

If modifications are made to an existing storage subsystem configuration, the Core Server and the Migration/Purge Server for the subsystem must be recycled.

15.2.5. Deleting a storage subsystem



Always contact HPSS support before deleting a subsystem definition. An improperly deleted subsystem can cause serious problems for an HPSS system.

It is critical that no files or directories exist within a storage subsystem before it is deleted. It is important to verify that all of the DB2 metadata tables associated with the storage subsystem being deleted are empty.

To verify that all files have been removed from the subsystem, perform the following steps:

1. Run the dump_acct_sum(7) utility on the subsystem. Be sure to specify the subsystem ID with the **-s** option as this utility defaults to subsystem 1. The output from the utility should indicate that zero files exist in the subsystem.
2. As a second verification do the following:
   a. Run the **db2** command line utility program.
   b. Connect to the subsystem database (for example, `connect to subsys1`).
   c. Set the schema (for example, `set schema hpss`).
   d. Issue the following SQL command:

      ```
      db2> select count(*) from bitfile
      ```

      The result of the command should indicate "0" rows in this table.

   e. Issue the following SQL command:

```
db2> select count(*) from nsobject
```

The result of the command should indicate "2" rows in this table.

3. If any of these checks gives an unexpected result, do not delete the subsystem. Contact HPSS support.



When deleting an existing storage subsystem, it is critical that all of the different configuration metadata entries described in the Storage Subsystem Configuration window for the storage subsystem be deleted. If this is not done, configuration metadata entries associated with the subsystem will become orphaned when the subsystem configuration itself is deleted. This situation is difficult to correct after the subsystem is deleted.

Once it has been verified that the storage subsystem contains no files or directories, deleting the subsystem may be accomplished by reversing the steps used to create the subsystem. These steps are listed in the order which they should be performed below. For information on each step, see the corresponding instructions under Creating a new storage subsystem.

1. Delete all storage resources assigned to the subsystem's Core Server. See Deleting storage resources.
2. Delete all filesets and junctions used to link the storage subsystem into the HPSS name space. See Deleting a junction.
3. Delete all storage class subsystem-specific thresholds associated with the storage subsystem being deleted. This is done by setting these thresholds back to the storage class "default" thresholds.
4. Delete all subsystem-specific migration and purge policy entries associated with the storage subsystem being deleted. See Deleting a migration policy and Deleting a purge policy for more information.
5. Delete all Core Servers, the Migration/Purge Server, and the DMAP Gateway Server residing within the storage subsystem being deleted. See Deleting a server configuration.
6. Delete the subsystem configuration.
7. Remove the database associated with the subsystem.

15.3. HPSS trashcans

The HPSS trashcan feature is similar to the trashcan feature found on many other systems. The purpose of all trashcan implementations is to provide a method to recover from deleting an entry that is still needed.

On and off

A notable difference from other trashcan implementations is HPSS's ability to turn the trashcan feature on and off globally. Sites not wishing to run with trashcans can easily configure the feature to be off. This is done by unchecking the Trashcans Enabled box in the Trashcan Settings section of

the *Global Configuration* window. (Refer to Global Configuration window)

The default HPSS configuration is to have trashcans turned off.

Sites choosing to run with trashcans turned off will not have the ability to recover deleted entries. With trashcans turned off, all deleted entries will be instantaneously and irrevocably lost.

When the trashcan feature is turned on, all deleted entries are moved into the `.Trash` directory for each user, where the entries will remain for a configurable amount of time. While entries are in the `.Trash` directory they can be recovered.

15.3.1. The trashcan feature is on

The remainder of this trashcan discussion and description assumes that the trashcan feature has been turned on.

Creating a .Trash directory

When an entry is deleted HPSS implicitly creates the trash directory (if the trash directory doesn't already exist), usually in the user's home directory. There is no need to explicitly create the `.Trash` directory.

If an entry is deleted from the root fileset, the `.Trash` directory is created in the user's home directory. The trash directory will be named:

```
~/.Trash
```

However if the entry being deleted does not reside in the root fileset, the trash directory will be created (if it doesn't already exist) in the root node of the fileset that contains the entry, and the trash directory will be named:

```
./.Trash/<user name>
```

The reason for putting entries into `.Trash` directories in the root nodes of filesets is because HPSS does not allow objects to be moved across fileset boundaries. This restriction is what prevents us from moving the entry into the `.Trash` directory in the user's home directory.

If user Joe Smith deletes `file6` from a fileset (other than the root fileset) that is accessed via the junction JunctionToFS,

```
rm /home/smith765/JunctionToFS/project3/file6
```

`file6` will be moved into the `.Trash` of the root node of the fileset as follows:

```
./.Trash/smith765/file6.<ObjectId>
```

Names

When entries are deleted they are moved into the `.Trash` directory. The `.Trash` directory is a flat, one-level directory. To prevent name collisions, the name of every object is made unique by appending the object's ObjectId to the end of its name. (Every HPSS object is assigned a unique number known as its ObjectId).

If user Joe Smith deletes file `Foo77` (which has ObjectId 54321) from his root fileset, it is moved into the `.Trash` directory with the following name

```
~/.Trash/Foo77.54321
```

What can be deleted?

Files, directories, junctions, and links can be put in the trash. Note that, as usual, the root nodes of filesets cannot be deleted by clients. Further note that, as always, directories must be empty before they can be deleted.

Immediate deletion

To cause an object to be immediately deleted, a user can delete a previously deleted object from the `.Trash` directory. There is no single-step permanent deletion method when the trashcan feature is enabled.

If user Joe Smith wants file `Foo77` to be permanently deleted immediately, this can be accomplished by deleting `Foo77`, then deleting it again from the trashcan with its appended ObjectId. If assistance is needed in obtaining the ObjectId, contact HPSS support.

```
rm  ~/OriginalPathTo/Foo77
rm  ~/.Trash/Foo77.54321
```

Object recovery

If user Joe Smith needs to recover `Foo77`, Joe can **cd** to the `.Trash` directory and move it out of the `.Trash`.

```
mv ./Foo77.54321 ../SafeDirectory
```

Another way to recover

In addition to moving an entry to recover it, there may be some interfaces that provide an undelete command which will move objects from the `.Trash` directory to their original location, assuming the original location still exists.

Finding entries in the trash

Because `.Trash` directories are single level directories, finding the desired entry among many

similarly named `.Trash` entries may prove to be difficult. For example, the `.Trash` directory could contain the following versions of the file `Foo77`:

```
Foo77.54321
Foo77.834239
Foo77.345678
```

To determine which of these files was the last one deleted, list the contents of the `.Trash` directory and review the time created. (Not all applications may support the flags shown in this example.)

```
ls -lt
```

Trash Time

The length of time that an object will remain in the trashcan before being permanently deleted is configurable and will probably be different at each HPSS site. The default length of time set by HPSS is 10 days. Note that this length of time is applied globally to all trashcans in the HPSS system. This length of time can be changed by an HPSS system administrator via the SSM's *Global Configuration* screen.

Note that the instant an object is deleted, its Trash Time starts counting. And whether the Trash Time is 60 seconds or 60 days, when the Trash Time is up, the object becomes eligible for permanent deletion and will likely be permanently deleted relatively quickly after becoming eligible, assuming the permanent deletion threads are able to perform as expected.

15.3.2. Trashcan operations for the administrator

There are some trashcan operations that can only be performed by the HPSS administrator. All of these trashcan configuration operations can be performed on the SSM's *Global Configuration* screen. Note that various trashcan-related counts and statistics can be viewed from the *Core Server Information* window.

TrashEnabled - enabling the trashcan feature

In the *Global Configuration* window, there is a check box for enabling the trashcan. Checking the box enables the trashcan, unchecking the box disables it. The box is unchecked by default. (Refer to Global Configuration window.)

Trash Unlink Eligible Time

Entries in trashcans are permanently deleted only after a specified amount of time has elapsed. **Trash Unlink Eligible Time** is used to specify that amount of time in seconds. For example, if **Trash Unlink Eligible Time** were to be set to "691200" then entries would be allowed to remain in trashcans for at least 8 days. The default setting is 864000 seconds which is 10 days.

Number of Trashcan Threads

Specifies the number of background threads that are to be employed permanently deleting entries from trashcans. This is a performance-tuning knob — for example, if the number of entries in trashcans seems to be high, or if there is a need to reclaim space consumed by trash entries, or if there is some other trashcan-related need, the administrator may wish to increase the number of trashcan-related background threads so that these tasks can be performed more quickly. On the other hand, if the number of trashcan background threads is taking up resources that might be needed elsewhere, the administrator may wish to decrease the number of threads.

If **Number of Trashcan Threads** is set to zero, HPSS interprets this as a request to no longer delete entries from `.Trash` (the NSTRASH database table). If the **Number of Trashcan Threads** is zero, deletions will continue to move entries into `.Trash` directories, but these entries will not be deleted until the **Number of Trashcan Threads** is set to a positive nonzero integer value.

Trash Incinerator Interval

The trashcan deletion threads run periodically in the background. This knob is provided to control this periodicity. **Trash Incinerator Interval** is specified in seconds. If **Trash Incinerator Interval** is set to "3600", the trashcan deletion threads will kick off every hour.

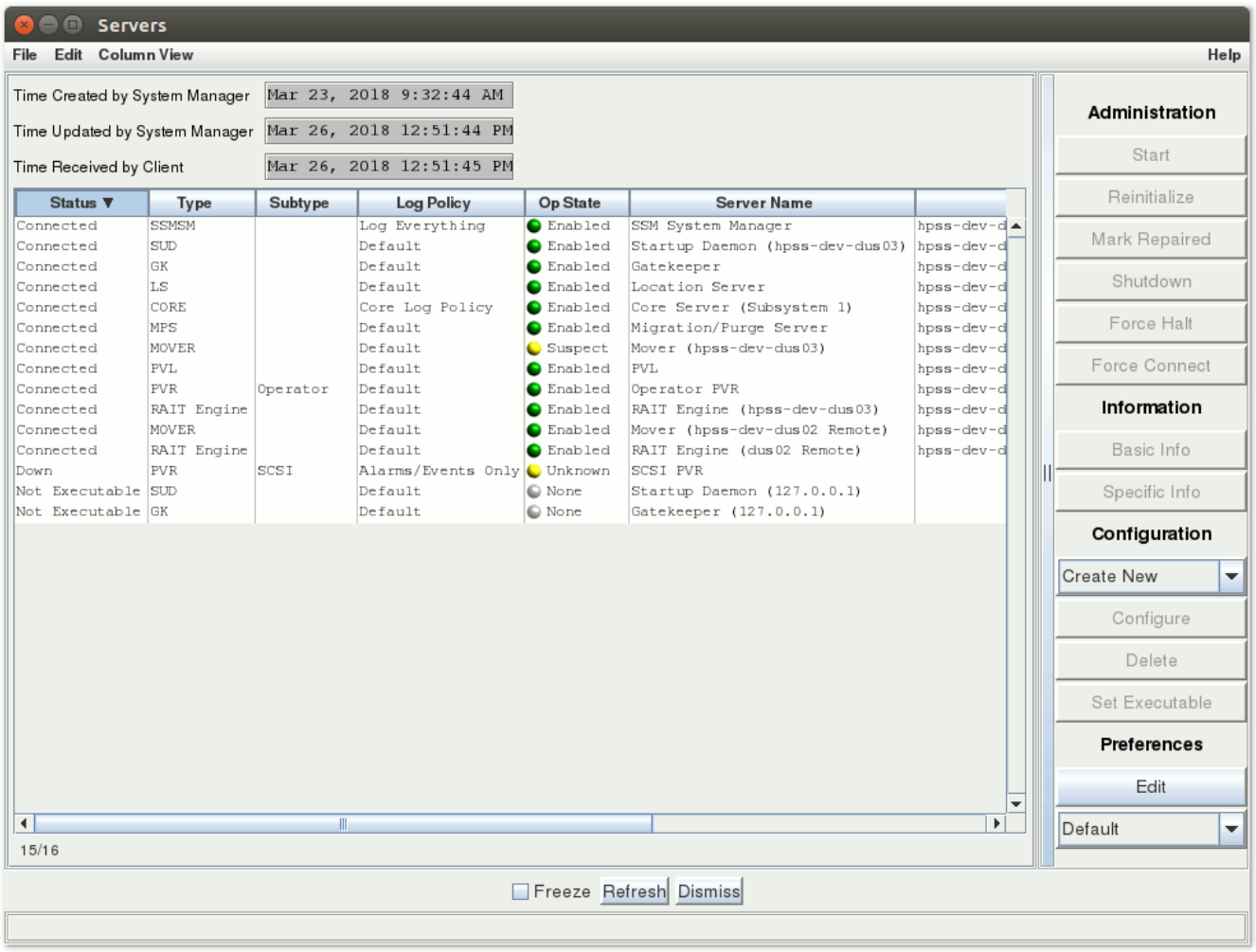
Trash Statistics Interval

The thread that gathers trashcan statistics runs periodically to gather statistics about all entries that are currently in trashcans. This knob controls how often this thread runs, and is specified in seconds. For example, if **Trash Statistics Interval** were to be set to "7200", the trashcan statistics thread would run every two hours. The administrator should be aware that the trashcan statistics thread might consume significant system resources when run on a system having a large number (millions) of entries in trashcans. In such cases the administrator might wish to increase the **Trash Statistics Interval** value so that the thread runs less often. Also, note that this thread will continue to run if the trashcan is not enabled.

Chapter 16. HPSS servers

Most HPSS server administration is performed from the SSM graphical user interface *Servers* list window. Each HPSS server has an entry in this list.

16.1. Server list



This window facilitates management of the configured HPSS servers. From this window, an HPSS server can be started, shut down, halted, reinitialized, and notified of repair. Once a server is up and running, SSM monitors and reports the server state and status. Information on running servers may also be viewed and updated via this window.

If multiple entries are selected in the server list when a server operation is invoked, the operation will be applied to all selected servers (although some servers don't support all operations).

The server entries displayed on the window can be sorted by each column category. To sort the server entries by status, for example, click on the Status column title. The actual display can vary greatly by the setting of window preferences. The Column View menu item can be used to select which columns of the table are visible, and Preferences can be used to further refine and filter which servers are visible. Preferences can also be saved and automatically reloaded by SSM when new sessions are started.

At times, the server list may update quite frequently with new Status or Op State information. If you select either of these columns for sorting, servers are likely to change their positions in the list rapidly. Sometimes this makes the list hard to use, in which case you should consider selecting a more static column for sorting or check the **Freeze** button to keep the list from updating.

Field descriptions

Server List

The main portion of the window which displays various information about each server.

ID

A unique numerical value assigned by SSM when each server starts. This value can change each time SSM is restarted. In the default preferences, this column is not shown.

Status

The server execution and connection status as determined by SSM. The reported status will be one of the following:

- `Connected` when the Server is up and running and communicating normally with SSM.
- `Up/Unconnected` when the Server is up and running (according to the Startup Daemon) but SSM cannot connect to it. Server cannot be completely controlled and monitored through SSM.
- `Down` when the Server is down. SSM can be used to start the server.
- `Indeterminate` when the server's state cannot be determined by SSM and the Startup Daemon is either not running or not connected to SSM.
- `Check Config` when SSM detected an incomplete or inconsistent configuration for the server. The server's configuration should be carefully reviewed to ensure that it is correct and complete. Check the *Alarms and Events* window and the HPSS log file to view SSM alarm messages related to configuration problems. This situation can be caused by:
  - A DB2 record required by the server is missing or inaccessible.
  - The principal name configured for the server does not match the HPSS_PRINCIPAL_* environment variable for the server's type.
- `Not Executable` when the server is configured as non-executable. SSM will not monitor the server's status.

  Note: the server could still be running. It may have been started outside the control of SSM, or it may have been running when its executable flag was changed.

- `Deleted` when the server configuration has been deleted. In the default preferences, this Status type is filtered off.

  Note: deleted servers will be removed from the list when SSM is restarted.

In addition to the above status values, the **Status** field also reports the transient status for the server as the result of the user request on the server as follows:

- `Starting···` when the server is being started.
- `Stopping···` when the server is being shut down gracefully.
- `Halting···` when the server is being forcibly halted.
- `Reiniting···` when the server is reinitializing.
- `Connecting···` when SSM is trying to establish a connection to the server.
- `Repairing···` when the server is repairing its states and statuses.

A server that is configured to execute and is running should have a `Connected` status. If its status is anything other than `Connected` (excluding the transient status values), one of the following actions should be taken:

- If the server status is `Up/Unconnected`: Monitor the server status closely for a few minutes. SSM will periodically try to establish a connection with the server. The **Force Connect** button can be used to speed this process.
- If the server status is `Down`: Use the **Start** button to start the server. The Startup Daemon will ensure that only one instance of the server is running.
- If the server status is `Indeterminate`: Verify whether the server is running using an operating system command such as **ps**. If the server is not running, start it. If the server is running, ensure that the Startup Daemon configured for the same node is running and has a connection to SSM. If the Startup Daemon is not running, start it using the `/opt/hpss/bin/rc.hpss` script on the appropriate node. Otherwise, use the **Force Connect** button to establish the connections for the server and the Startup Daemon. If this does not correct the server's status, review the *Alarms and Events* window to search for problems that the server and SSM may have reported. In addition, review the HPSS logs for the server's and SSM's log messages to help determine the problems.

SSM periodically polls the execution and connection status of the servers and updates the **Status** fields when there are changes. The rate of these updates depends on the client's refresh rate (see the hpssgui(7) or hpssadm(7) man pages for more details).



If a server is configured as executable but is not running, SSM will treat it as an error. Therefore, if a server is not intended to run for an extended period, its Executable flag should be unchecked. SSM will stop monitoring the server and will not report the server-not-running condition as a critical error. This will also help reduce the work load for SSM.

Type

Indicates the type of the server in acronym form. Possible values include CORE, GK, LS, MPS, MOVER, PVL, various PVR values, SSMSM, and SUD. See the glossary for the meanings of these acronyms.

Subtype

Indicates the subtype of the server, sometimes in acronym form. Not all servers have subtypes. Possible values (particularly PVR subtypes) include SCSI, STK, and Operator. See the glossary for the meanings of these acronyms.

Subsystem

The name of the storage subsystem to which the server is assigned. Only servers of type Core and MPS should have a subsystem name. For all other server types this column should be blank.

Op State

The operational state of the server as reported by the server itself:

- `Enabled` indicates the server is operating normally.
- `Disabled` indicates the server is not operational, usually due to a shutdown or halt request.
- `Suspect` indicates the server may have a problem.
- `Minor` indicates the server encountered a problem that does not seriously affect the HPSS operation.
- `Major` indicates the server encountered a problem that may seriously impact the overall HPSS operation.
- `Broken` indicates the server encountered a critical error and shut itself down.
- `Unknown` indicates the server's state is unknown to SSM. SSM should be able to obtain an Operational State for the server but cannot because SSM cannot communicate with it, and the server has not set its state to `Disabled` or `Broken`. The server may or may not be running. The reason may simply be that the server is marked executable but is not running.
- `Invalid` indicates SSM has obtained an Operational State from the server, but its value is not recognized as a valid one. This may indicate a bug in the server or a communications problem which is garbling data.
- `None` indicates the server does not have an Operational State (because the server is not executable, for example), or its Operational State is, by design, unobtainable.

SSM will periodically poll the server's operational state and update the **Op State** field in the *Servers* window when there are any changes. The rate of the polling is controlled by the SSM client list refresh rate (see the hpssgui(7) or hpssadm(7) man pages).

Server Name

The descriptive name of the server.

Host

The name of the host on which the server is running. The **Host** is obtained by looking at the RPCs that are sent from the server back to the System Manager. If the server is not running, then this field will be blank. In most cases, this field will match the **Execute Host**; sometimes, in the case of multi-homed systems, this field won't match the **Execute Host**.

Execute Host

The **Execute Hostname** field from the server's basic configuration record. This field is intended to specify the hostname on which the server is supposed to run; however, no checking is done to verify if the server is actually running on the specified host. This field is only used by the SSM to locate the Startup Daemon that manages this server. The field

displayed must match exactly the **Execute Hostname** field specified in the configuration record of the startup daemon that is to manage this server.

UUID

The universal unique identifier for the server. In the default preferences, this column is not shown.

Administration buttons

This group of buttons allows you to perform administrative operations on the selected servers. All of the buttons are disabled unless one or more servers are selected and the operation is applicable to at least one of the selected servers.

After clicking one of these buttons, you will usually be prompted to confirm your request before it is carried out. If you make an obviously invalid request, such as asking to start a server which is not executable, SSM will tell you about it, but otherwise no harm is done. You could, for example, select a range of servers and ask to start them all, knowing that some of the selected servers are already running. Those that are not running will be started, but nothing will be done to those that are already running.

The status bar at the bottom of the window displays a message when the requested operation has begun. For information on the success or failure of the operation, monitor the Status and Op State columns for the server.

Start

Start the selected servers. The System Manager will notify the Startup Daemon to start the selected servers.

Reinitialize

Send a "reinitialize" command to the selected servers. The Startup Daemon does not support reinitialization. If a server does not support reinitialization, the button will not be enabled.

Mark Repaired

Clear displayed error status in the selected servers. Sometimes server states such as the Operational State will continue to indicate an error condition after the cause of the error has been fixed. If so, you can use the **Mark Repaired** button to clear its error states. Note that this does not do anything in hardware or software to actually repair an error; it just asks the server to clear its error condition. If you mark a server repaired when it still has a problem, the error states will be cleared but may quickly return.

Shutdown

Command the selected servers to shut down. This command should be the first you use to shut down servers. After issuing this command, wait a minute or two for the server to complete the shutdown process. Some servers shut down very quickly after receiving the command, others, particularly the Core Server, may require two minutes or more to shut down. During this time the server is attempting to finish work it has started, while rejecting new work. Be patient; watch the *Alarm and Events* window for messages indicating the server has terminated.

Force Halt

Command the selected servers to stop immediately. This should be done only if a shutdown request has failed to stop the servers, or if the intention is to shut servers down as quickly as possible. This request will cause a SIGKILL signal to be sent to the selected servers. This command is meant to be used as a last resort if a server hangs up or otherwise won't respond to the Shutdown command.

Force Connect

Request the System Manager to immediately attempt to connect to the selected servers. The System Manager routinely attempts to connect to any unconnected servers; using this button will simply cause the next attempt to occur right away, instead of after the normal retry delay.

Information buttons

These buttons allow you to open information windows for servers. They will be enabled only when the selected server supports a particular information window.

Since the requested information is obtained by calling the selected server, the server must normally have a Connected status for the request to succeed.

Basic Info

Opens the *Basic Server Information* window for the selected server.

Specific Info

Opens the type-specific server information window for the selected server.

Configuration buttons

These buttons allow you to start server configuration tasks.

Create New

Allows an administrator to configure a new server by selecting a server type and then filling in a new server configuration. This control is actually not a button, but a drop-down list, allowing the administrator to select the type of new server to be created. It is always enabled.

Configure

Opens a configuration window for each selected server.

Delete

Deletes the selected servers from the system. A confirmation dialog will appear to confirm the action. Before deleting a server, see the warnings and considerations in Deleting a server configuration.

Set (Non-)Executable

Sets or unsets the Executable configuration flag for each selected server.

16.2. Server configuration

The following HPSS servers should be configured. A Gatekeeper is required only if the site wishes to do gatekeeping or account validation. A RAIT Engine is required only if the site uses RAIT volumes. See the HPSS servers section for a description of the purpose of each server:

- Core Server
- Migration/Purge Server
- Gatekeeper (optional)
- Physical Volume Library
- Physical Volume Repository
- Mover
- RAIT Engine (optional)
- Storage System Manager
- Startup Daemon (on each host where an HPSS server will be executing)

The fields of the *Server Configuration* window are divided into the following sections. The Basic Controls section is at the top of the window and the other sections are on individual tabs:

Basic Controls

Server identification and type information.

Execution Controls

Information required to properly control the server's execution.

Interface Controls

Information required for communication between the server and other HPSS servers and clients.

Security Controls

Security settings.

Audit Policy

Object event audit settings.

Specific

Server type-specific settings (only for some server types).

The server-specific configuration section contains configuration parameters specific to a particular server type. Not every type of server has a specific configuration section. The following types of servers have a server-specific configuration section in the *Server Configuration* window:

- Core Server
- Gatekeeper
- Migration/Purge Server

- Mover
- Physical Volume Repository
- RAIT Engine

Details about the specific configuration section and the additional configuration required for each of these server types are described in:

Core Server-specific configuration

Gatekeeper-specific configuration

Migration/Purge Server (MPS)-specific configuration

Mover-specific configuration

Physical Volume Repository (PVR)-specific configuration

RAIT Engine-specific configuration

- Details about all of the other sections on this window, which apply to all server types, are described in Common server configuration.

To view the *Server Configuration* window for an existing server, bring up the *Servers* list window and select the desired server. Then click the **Configure** button to bring up the configuration window for that server.

To modify a server configuration, bring up the server's configuration window, modify the desired fields, and click the **Update** button. Some configuration fields may be set by the administrator only at server creation time. These fields will be editable and settable when creating new servers, but will become display-only fields (not editable or settable) when looking at existing configurations. For the modified configuration data to be recognized by the server, if the setting changed allows, the change should be applied by clicking the **Apply Config** button on the *HPSS Health and Status* screen. Otherwise, the server must be reinitialized (if supported) or restarted.

The Startup Daemon does not support "Apply Config" or reinitialization and must be restarted for configuration changes to take effect.

There are two ways to create a new server:

1. To create a new server using standard defaults as a starting point, open the *Servers* list window, click **Create New** and select a server type. This opens a *Server Configuration* window with default values for the selected server type. Make the desired customizations to the window and click the **Add** button.
2. To create a new server that has attributes similar to those of an existing server, select the existing server, click **Configure** and then select **Clone (partial)**. The **Update** button will be changed to **Add**. Make the desired customizations to the values in the window and click the **Add** button to save the new configuration.

There are two ways to delete a server. Before deleting a server, see the warnings and considerations in Deleting a server configuration.

Then, to delete the server:

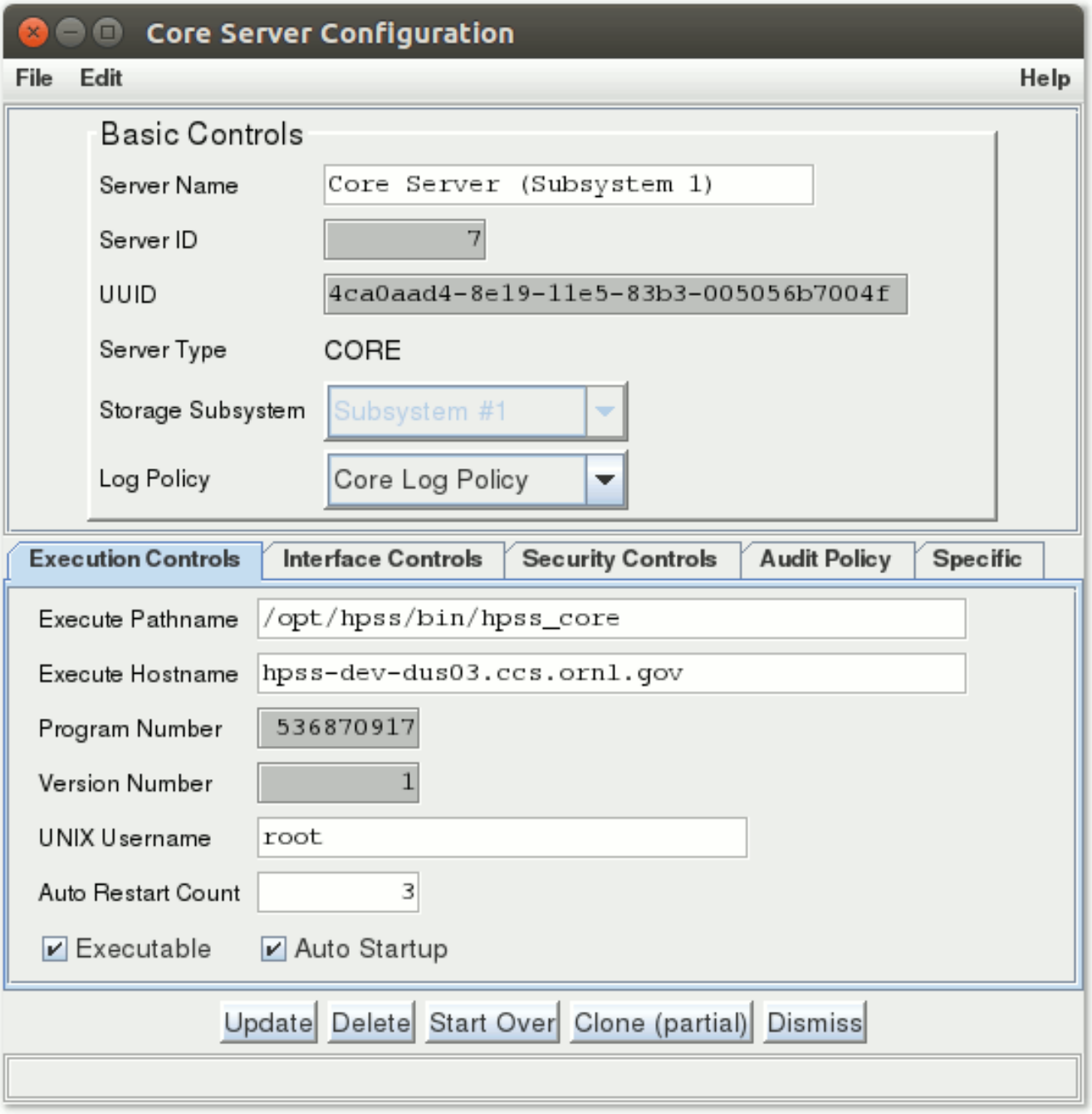
1. From the *Servers* list window, select the server and click the **Delete** button, or
2. Bring up the *Server Configuration* window and click the **Delete** button.

In both cases you will be prompted to confirm the deletion of the server.

16.2.1. Common server configuration

These sections of the *Server Configuration* window are common to all servers.

16.2.1.1. Basic controls



The Basic Controls section of the *Server Configuration* window is common to all servers. In the example window above, the server displayed is a Core Server.

Field descriptions

Server Name

A unique descriptive name given to the server. Ensure that the **Server Name** is unique. A server's descriptive name should be meaningful to local site administrators and operators, in contrast to the server's corresponding UUID, which has meaning for HPSS. For HPSS systems with multiple subsystems it is very helpful to append the subsystem ID to the **Server Name** of subsystem-specific servers. For instance, "Core Server 1" for the Core Server in subsystem 1.

Server ID

A unique value automatically generated and displayed for servers.

UUID

A universal unique identifier that identifies a server to HPSS. A unique value is automatically generated and displayed for new servers.

Server Type

The type of the HPSS Server.

Server Subtype

The subtype of the selected server. This field is only used by the PVRs to specify the type of PVR (for example, STK).

Storage Subsystem

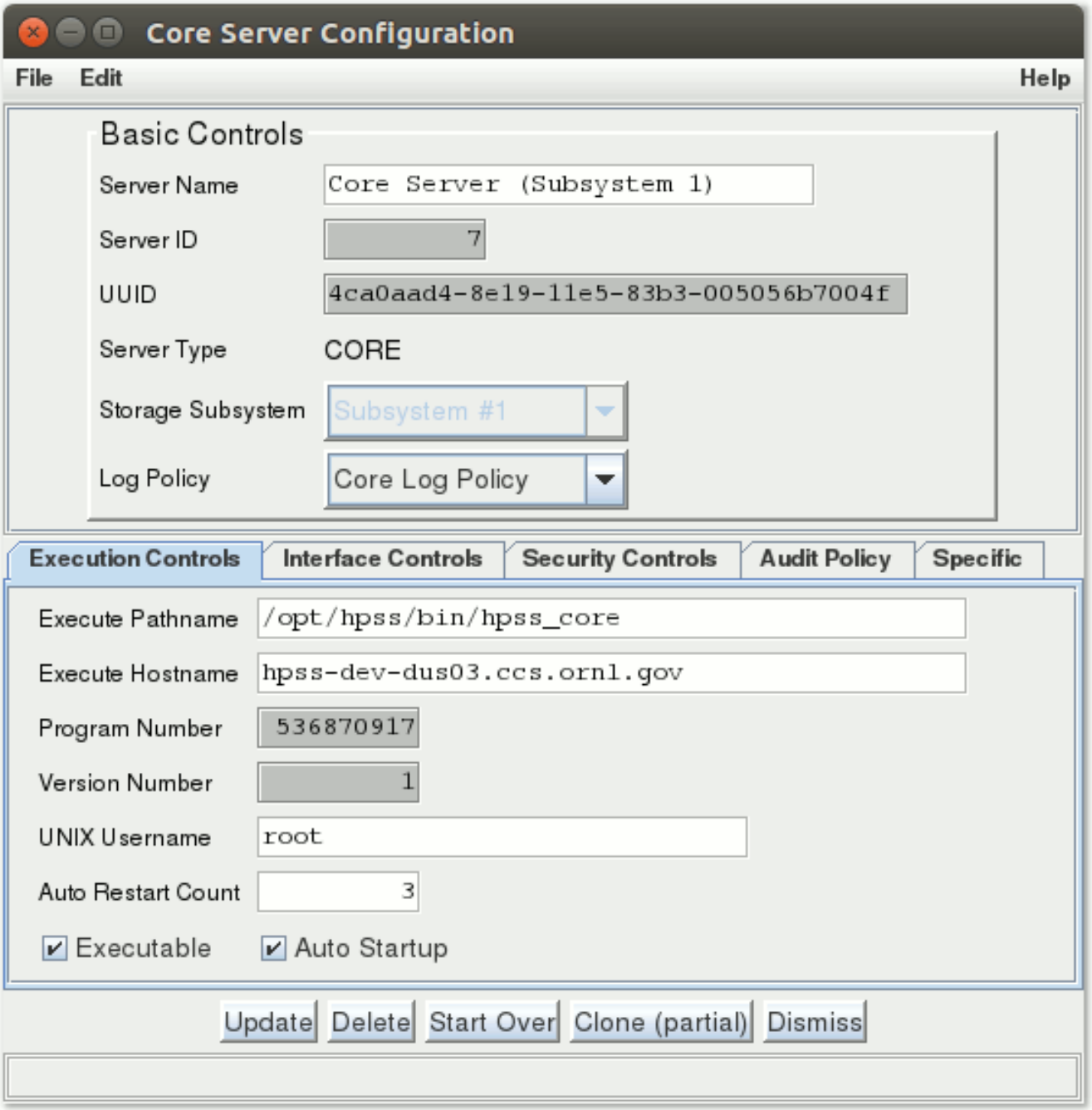
The name of the HPSS storage subsystem to which this server is assigned. This field is required for the Core Server and Migration/Purge Server. For all other servers this field is not displayed. The following rules apply to this field:

- The storage subsystem configuration must exist before configuring Core Servers or Migration/Purge Servers.
- Core Servers and Migration/Purge Servers must have a valid storage subsystem set in this field before SSM will allow the server configuration to be created.
- No more than one Core Server and one Migration/Purge Server can be assigned to any one subsystem.
- This field can only be modified in Add mode.

Log Policy

The log policy used to specify which log message types will be sent to syslog for this server. See Log policies for a description of the *Logging Policies* window, for detailed definitions of each log message type, and for information on the system default logging policy.

16.2.1.2. Execution controls



The **Execution Controls** section of the *Server Configuration* window is common to all servers. In the example window above, the server displayed is a Core Server.

Field descriptions

Execute Pathname

The UNIX file system path name to a server's executable image file. This file must reside on the node specified by **Execute Hostname.** Use the full UNIX path name; otherwise, the Startup Daemon will try to start the file out of the current working directory of the Startup Daemon.

Execute Hostname

This is the hostname of the node on which the server will execute. It must match the **Execute Hostname** of the Startup Daemon that is to manage this server. For most servers, setting this field is straightforward, but for remote Movers, this indicates the node on which the Mover administrative interface process runs (not the node where the remote Mover process runs). Note

that if the **Execute Hostname** is changed, it is likely that the RPC program number will change as well. If the server affected is the System Manager, the SSM configuration file, `ssm.conf`, must be regenerated or edited.

Program Number

The RPC program number. This value must be unique within the node in which the server runs. The administrator cannot override the default program number value when creating or modifying a server configuration.

Version Number

The RPC version number. The administrator can not override the default version number value when creating or modifying a server configuration.

UNIX Username

The UNIX user name under which the server will run. The name must be registered in the local UNIX authentication database (for example, `/etc/passwd`) or the HPSS local password file. The Startup Daemon will use this user ID when starting the server. If there is no such user, the Startup Daemon will not be able to start the server.

Auto Restart Count

Maximum number of automatic restarts allowed per hour. See Automatic server restart for more details.

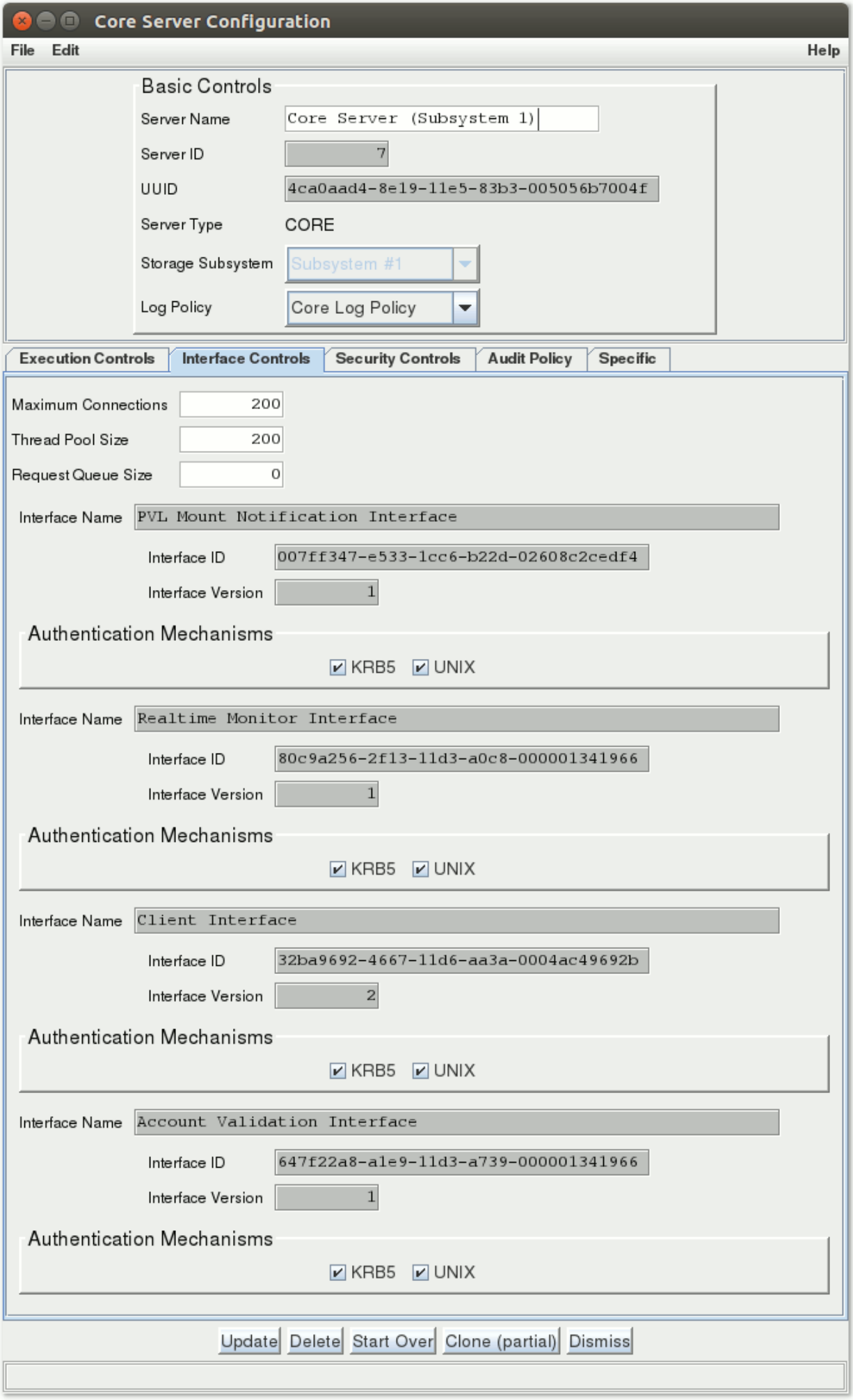
Executable

A flag that indicates whether an HPSS server can be executed by SSM.

Auto Startup

A flag that indicates whether SSM should attempt to start the server upon startup of SSM. The Startup Daemon must be started before SSM for this option to work.

16.2.1.3. Interface controls


Core Server Configuration
File
Edit
Help
Basic Controls
Server Name
Core Server (Subsystem 1)
Server ID
7
UUID
4ca0aad4-8e19-11e5-83b3-005056b7004f
Server Type
CORE
Storage Subsystem
Subsystem #1
Log Policy
Core Log Policy
Execution Controls
Interface Controls
Security Controls
Audit Policy
Specific
Maximum Connections
200
Thread Pool Size
200
Request Queue Size
0
Interface Name
PVL Mount Notification Interface
Interface ID
007ff347-e533-1cc6-b22d-02608c2cedf4
Interface Version
1
Authentication Mechanisms
KRB5
UNIX
Interface Name
Realtime Monitor Interface
Interface ID
80c9a256-2f13-11d3-a0c8-000001341966
Interface Version
1
Authentication Mechanisms
KRB5
UNIX
Interface Name
Client Interface
Interface ID
32ba9692-4667-11d6-aa3a-0004ac49692b
Interface Version
2
Authentication Mechanisms
KRB5
UNIX
Interface Name
Account Validation Interface
Interface ID
647f22a8-a1e9-11d3-a739-000001341966
Interface Version
1
Authentication Mechanisms
KRB5
UNIX
Update
Delete
Start Over
Clone (partial)
Dismiss

The **Interface Controls** section of the *Server Configuration* window is common to all servers. In the example window above, the server displayed is a Core Server.

Field descriptions

Maximum Connections

The maximum number of clients that this server can service at one time. This value should be set based on the anticipated number of concurrent clients. Too large a value may cause unwarranted memory usage. Too small a value will mean that some clients may not be able to connect. This value should be chosen such that it is equal to between one and two times the number of clients that can be initiating connections simultaneously.

Thread Pool Size

The number of threads this server spawns to handle client requests. If necessary, the default values can be changed when defining servers. Too large a value may consume server memory for no purpose. Too small a value could mean that some client requests are rejected or don't get serviced in a timely manner. The Thread Pool Size should be equal to or larger than the number of expected simultaneous requests.

Request Queue Size

The maximum number of requests that can be queued waiting for request threads. If the workload increases so that this value is exceeded, requests will be rejected by the server rather than queued for processing. A value of zero means to use the default queue size of 20.

Note: See Tuning the System Manager RPC thread pool and request queue sizes for information on tuning the RPC thread pool and request queue sizes for the System Manager.

The following three interfaces fields provide the information required for the server to communicate with other HPSS servers and clients over the HPSS RPC network interface. Each server type is configured with one or more interfaces. With the exception of the Authentication Mechanisms, the administrator cannot override the default values for each interface when creating or modifying a server configuration. Each interface consists of the following fields:

Interface Name

The descriptive name of the interface.

Interface Id

The UUID that identifies this interface.

Interface Version

The interface version number.

Authentication Mechanisms

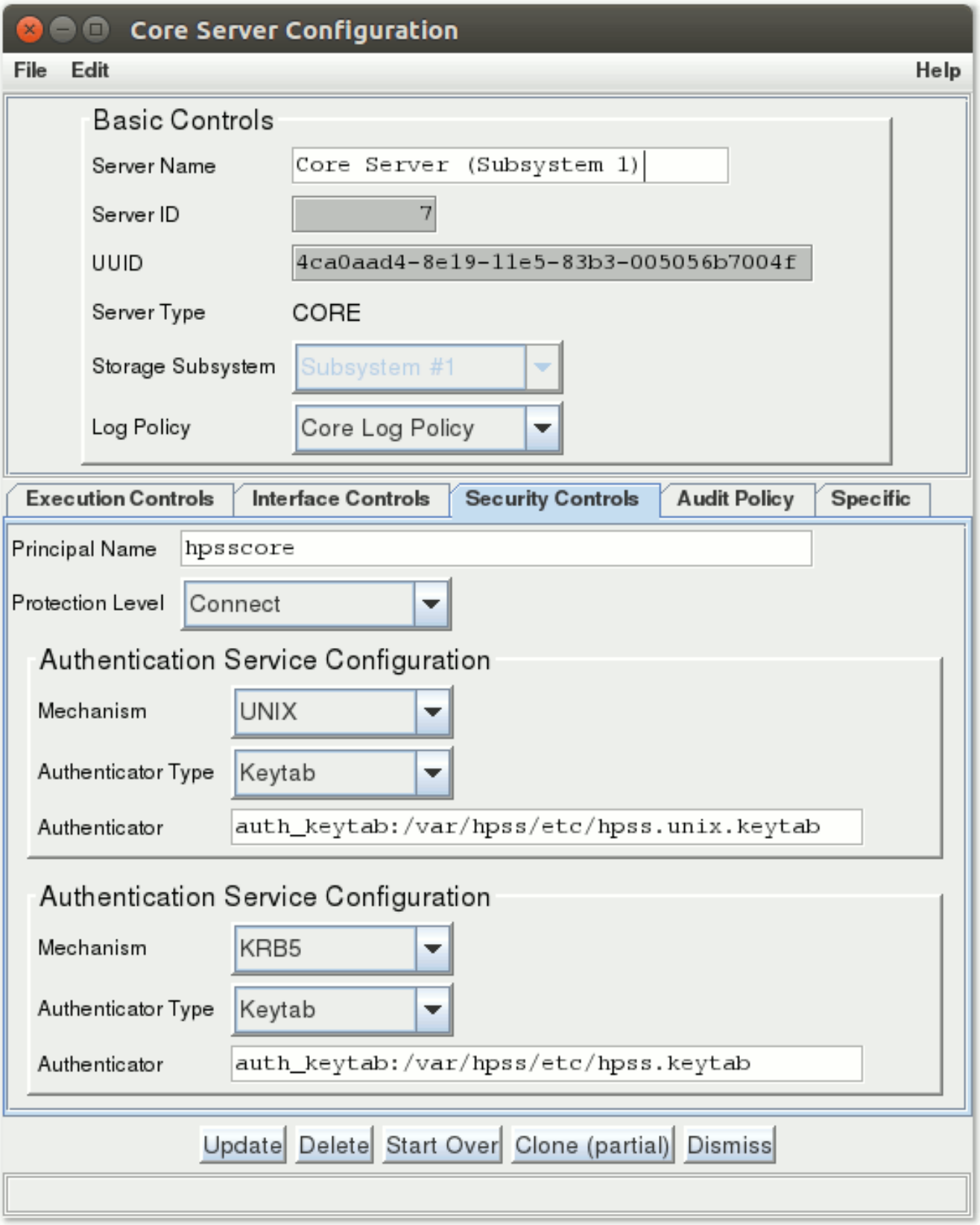
The authentication mechanisms from which the interface will accept credentials. One or both of the mechanisms can be checked (at least one should be checked or that interface will be unusable). Each interface supports the following authentication mechanisms:

- KRB5 indicates that the interface will support Kerberos 5 authentication.

- UNIX indicates that the interface will support UNIX authentication.

16.2.1.4. Security controls

The security controls fields define the settings for authenticated communication between the server and other HPSS servers and clients.



The **Security Controls** section of the *Server Configuration* window is common to all servers. In the

example window above, the server displayed is a Core Server.

Field descriptions

Principal Name

The name of the principal the server will use to authenticate.

Protection Level

The level of protection that will be provided for communication with peer applications. The higher the level of protection, the more encryption and overhead required in communications with peers. The levels, from lowest to highest, are as follows:

`Connect`

Performs authentication only when the client establishes a connection with the server.

`Packet`

Ensures that all data received is from the expected client.

`Packet Integrity`

Verifies that none of the data transferred between client and server has been modified.

`Packet Privacy`

Verifies that none of the data transferred between client and server has been modified and also encrypts the data transferred between client and server.

Authentication Service Configuration

Each server can support up to two Authentication Services. The following fields are used to define each authentication service configured for a server.

Mechanism

The authentication mechanism to use when passing identity information in communications to HPSS components.

- `KRB5` indicates that the server will use Kerberos 5 authentication.
- `UNIX` indicates that the server will use UNIX authentication.
- `Not Configured` indicates that an authentication service has not been configured for this slot. At least one of the authentication service slots must be configured.

Authenticator Type

The type of authenticator specified in the **Authenticator** field. The types are:

- `Not Configured` indicates that an authenticator has not been configured for this slot. If a mechanism is specified, an authenticator type must also be specified.
- `None` indicates no authenticator is supplied for this mechanism. This is appropriate for UNIX authentication if no keytab is used. The server's credentials will be its current UNIX identity.
- `Keytab` indicates that the authenticator is the path to a keytab file. For Kerberos

authentication this is a keytab file created with Kerberos utilities. For UNIX authentication this is a keytab file created with the hpss_unix_keytab(7) utility. See its man page for details. Each server can have its own keytab file, or all the servers can share a single keytab file. It is recommended that one keytab file be used for all of the servers on any given host.

Authenticator

The argument passed to the authentication mechanism indicated by the **Authenticator Type** configuration variable and used to validate communications. If it is a keytab, the server must have read access to the keytab file. Other access permissions should not be set on this file or security can be breached. For the Not Configured or None values of the **Authenticator Type**, this field can be left blank.

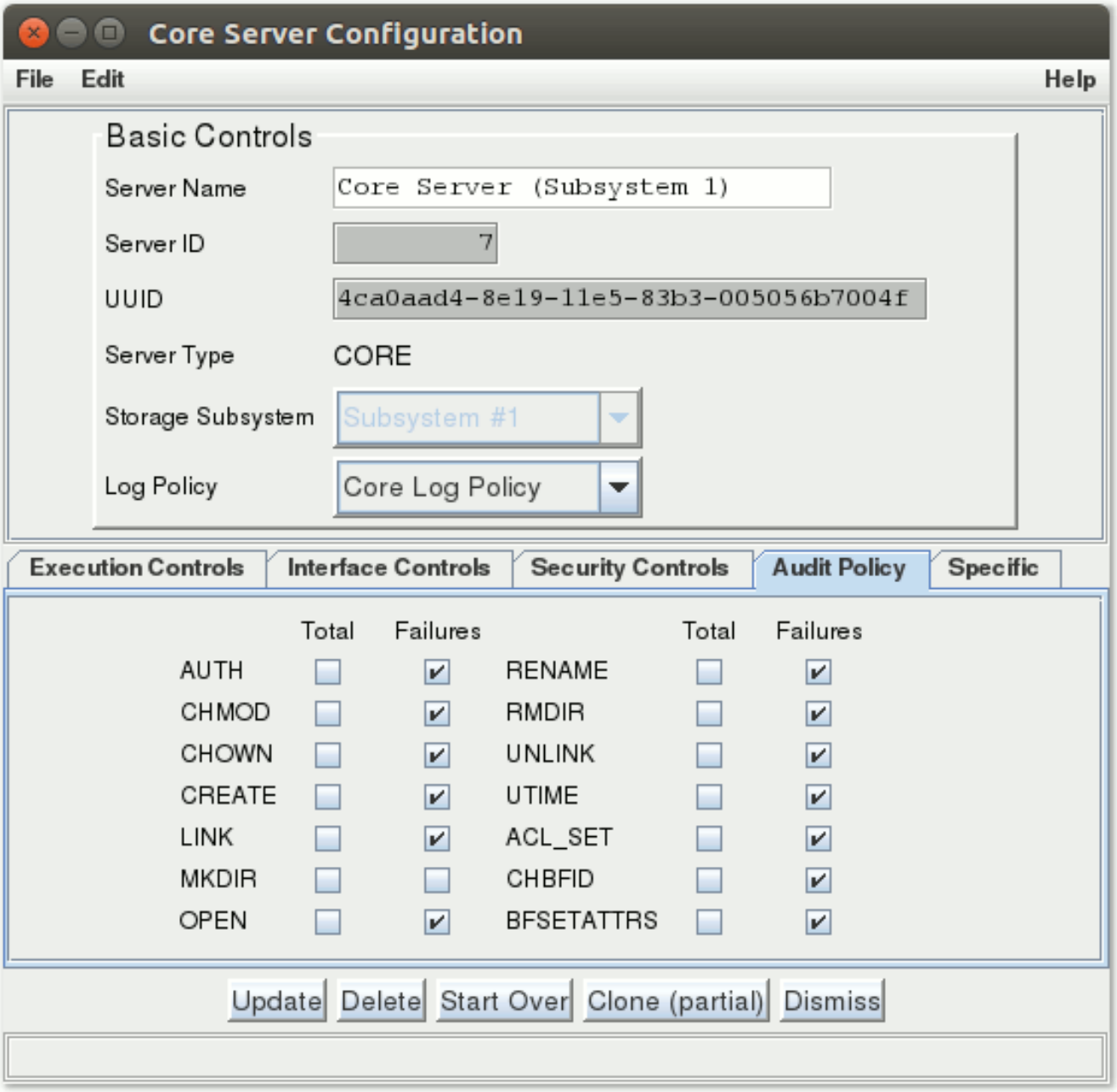
16.2.1.5. Audit policy

HPSS security provides a log for certain security-related events such as a change in an object's permissions, a change in an object's owner, a change of an object's name, a failed attempt to access an object, and several others (see *Field descriptions* below for a complete list). A server's Audit Policy controls which of these events trigger log messages. It can be configured from the **Audit Policy** tab of the *Server Configuration* window. It is possible to request that only failures or both successes and failures be logged.

For each audit event type, selecting **Failures** causes only failures to be logged, selecting **Total** logs every audit event of that type, and selecting neither causes nothing to be logged. Selecting both has the same effect as selecting **Total**.

Currently the only server which generates security object event records other than AUTH is the Core Server.

For more information on auditing, see the Security audit section.



The **Audit Policy** section of the *Server Configuration* window is common to all servers. In the example window above, the server displayed is a Core Server.

Field descriptions

AUTH

Authentication events.

CHMOD

Core Server object permission events.

CHOWN

Core Server object owner events.

CREATE

Core Server creation events.

LINK

Core Server hard link creation events.

MKDIR

Core Server directory creation events.

OPEN

Core Server bitfile open events.

RENAME

Core Server object rename events.

RMDIR

Core Server directory remove events.

UNLINK

Core Server object delete events.

UTIME

Core Server bitfile time modified events.

ACL_SET

Core Server access control list modification events.

CHBFID

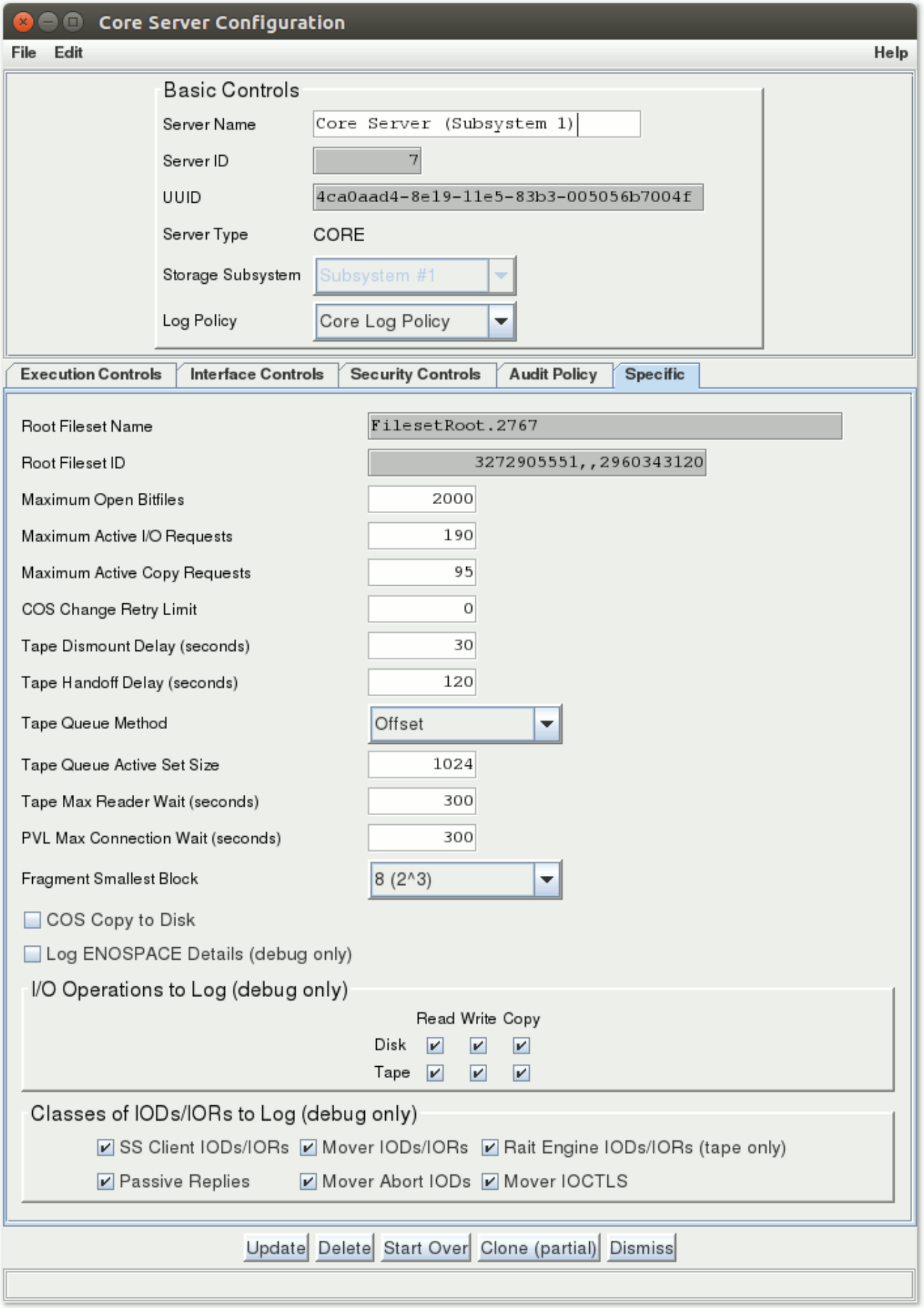
Core Server change bitfile identifier events.

BFSETATTRS

Core Server set bitfile attribute events.

16.2.2. Core Server-specific configuration

A separate Core Server must be configured for each storage subsystem.



The **Specific** tab of the *Core Server Configuration* window allows you to view and update the type-specific configuration for the Core Server.

The settings on this screen can be changed while the Core Server is running by changing the value on the screen, clicking the **Update** button to update the metadata, and then clicking the **Apply Config** button on the *HPSS Health and Status* screen to reinitialize the appropriate Core Server. The Core Server rereads the metadata and changes its internal settings. See Reinitializing a server for more information.



Changes to switches under **I/O Operations to Log (debug only)** and **Classes of IODs/IORs to Log (debug only)** take effect immediately after clicking the **Update** button and do not require server reinitialization.

Field descriptions

Root Fileset Name

Core Servers can create and support multiple filesets, but an initial fileset is required for each Core Server. The **Root Fileset Name** designates which of these filesets will be used by the Core Server to resolve the pathname "/" (forward slash) in the subsystem. Other filesets served by this Core Server, or from other Core Servers in the HPSS system may be joined to this fileset by junctions.

Root Fileset ID

The Fileset ID of the fileset named in **Root Fileset Name**.

Maximum Open Bitfiles

The maximum number of bitfiles that can be open simultaneously. This value is augmented by the number of open files required for tape aggregation. See Core Server max open files configuration for more information.

Maximum Active I/O Requests

The maximum number of simultaneous I/O requests allowed. (See Core Server I/O limit configuration for a detailed description of how this field interacts with other configuration fields.)

Maximum Active Copy Requests

The maximum number of simultaneous copy requests allowed. (See Core Server I/O limit configuration for a detailed description of how this field interacts with other configuration fields.)

COS Change Retry Limit

This is the maximum number of attempts that will be made to change the Class of Service of a file. If set to "0", COS change will continue to be attempted until it is successful. If a positive value is provided, the COS change request will be dropped after it has failed the configured number of times.

Tape Dismount Delay (seconds)

The amount of time, in seconds, a mounted tape volume will remain idle before being dismounted by the Core Server. Larger values may reduce undesirable tape dismount/remount events, at the expense of lower tape drive utilization.

Tape Handoff Delay (seconds)

The amount of time, in seconds, a mounted tape will be held in a client's session before become eligible to be handed off to another client that wishes to use the tape.

Tape Queue Method

This field may be set to either `FIFO` or `Offset`. If set to `FIFO`, the Core Server will process tape requests per VV in a FIFO fashion without doing any intelligent ordering. This somewhat mimics legacy behavior and can be useful for sites which want to closely manage their own ordering. If set to `Offset`, the Core Server will attempt to reorganize the VV session queue in order to improve performance. If available, this may use drive technologies such as Recommended Access Ordering (RAO).

If a file called `nodrive` exists in the HPSS tmp folder (HPSS_PATH_TMP), its presence will disable RAO-based sorting of tape requests and instead use strict offset ordering. If a file called `noschedule` exists in this folder it will disable all scheduling, resulting in a FIFO ordering for tape requests from that point until the file is removed. Note that this will not impact any existing schedules.

Individual VVs may also be toggled to FIFO scheduling using the **FIFO** check box on the VV information window. See the Core Server Tape Volume information window for more information.

Tape Queue Active Set Size

Number of sessions which may be scheduled together. This yields a compromise between scheduling efficiency and potential wait times since this size is the maximum number of items within that set that may be scheduled ahead of any other request in that set. This may also be limited by the hardware limitations of Recommended Access Ordering (RAO) if the drive a tape is mounted on supports RAO.

Tape Max Reader Wait (seconds)

This is the amount of time, in seconds, that readers of a tape may spend waiting on writers before the scheduling priority shifts to prefer readers over writers for some period of time. There is a minimum of five minutes between reader/writer priority changes. Setting this to zero disables preference switching and will cause writers to always be preferred.

PVL Max Connection Wait (seconds)

The amount of time, in seconds, the Core Server will wait to connect to the PVL before declaring an error in pending PVL jobs.

Fragment Smallest Block

Fragment Smallest Block sets the boundary used to determine where to remove excess clusters from allocations that have excess space. The smallest block returned to the free space map from the excess space at the end of an extent will be this size or larger.

COS Copy to disk

If ON, all copy operations associated with COS changes will be directed to disk if the hierarchy has a disk storage class at its top level, and will subsequently be migrated down the hierarchy. If OFF, COS changes are not automatically copied to the disk level. They will instead go to the top-

level tape storage class and then migrate down from there.

Utilizing COS Copy to disk allows migration to occur from that disk COS according to the disk's migration policy with features such as ordered migration and tape aggregation. Otherwise, files are written individually to tape for each COS stream. A large number of COS change streams could cause data to be spread out over a large number of tape volumes. Having this feature **off** is generally inefficient for small files or a large number of COS change streams, and could result in poor COS change performance or recall.

COS Copy to disk is recommended for systems that contain significant numbers of small files.

Log ENOSPACE Details (debug only)

If ON, the Core Server logs detailed debug class log messages about disk and tape allocation requests that end in HPSS_ENOSPACE. In normal operation, these details are too voluminous to log economically, so they are not logged by default. But if the cause of an ENOSPACE condition is not evident, setting this switch will enable logging of all of the details. Because of the volume of information logged, leaving this switch on for long periods of time is not recommended.

I/O Operations to Log (debug only), Classes of IODs/IORs to Log (debug only)

These switches control optional logging in the Core Server. The Core Server always logs IODs and IORs when I/O operations fail, but does not log IODs and IORs for successful operations. These switches allow the administrator to turn on logging of IODs and IORs in successful tape and disk I/O operations for diagnostic purposes.

Normally these switches should be off to avoid logging large amounts of information that won't be useful. Changes to these switch settings in the managed object take effect immediately. Reinitialization of the server is not necessary.

16.2.2.1. Additional Core Server configuration

Certain aspects of the operation of the Core Server can be controlled by setting environment variables. The server uses built-in default values for these settings, but if the environment variables can be found in the server's environment, the server uses those values. The following is a list of the variables and the aspects of the server's operation they control. Since the environment is common to all subsystems, all Core Servers in an HPSS installation are subject to these values.

HPSS_MAX_NUM_OPEN_FILES_FOR_AGGR

Adjusts the maximum number of files that may be open by the Core Server when it is assembling migration file aggregates. If this variable is not present in the environment, the server's default value is 20000. See Core Server max open files configuration for more information.

HPSS_SPEC_USER_LIST_LENGTH

Adjusts the length of the Core Server's internal list of users who have delete only permission in AUTHZACL table. If this variable is not present in the environment, the server's default value is 8.

HPSS_CORE_ATIME_POLICY

Selects the algorithm the Core Server will use whenever it computes the atime. There are three possible algorithms: CORE_ATIME_POLICY_NOATIME, CORE_ATIME_POLICY_RELATIME, and

CORE_ATIME_POLICY_STRICTATIME. CORE_ATIME_POLICY_NOATIME indicates that the atime is not to be calculated or set. The CORE_ATIME_POLICY_RELATIME value indicates that the Unix Relative atime algorithm is to be used when computing the atime. The CORE_ATIME_POLICY_STRICTATIME value indicates that the Core Server is to use the traditional Unix algorithm when computing the atime. Note that the traditionial Unix algorithm (CORE_ATIME_POLICY_STRICTATIME) can cause performance issues and these performance issues are the reason that the Relative atime algorithm was developed. Further note that the CORE_ATIME_POLICY_RELATIME is recommended and is the default algorithm used by the Core Server when computing the atime.

HPSS_CORE_REPACK_OUTPUT

Enables or disables the repack output segregation feature. If this environment variable is not present in the environment, or has the value "on", files moved from one tape virtual volume to another by repack(7) will be written to tapes that contain only files written by the repack(7). If this environment variable has the value "off", files moved from one tape virtual volume to another by repack(7) may be mixed on output tapes with files written by the migration subsystem, or directly to tape by users. When set to "on", this feature can be ignored on a per-repack basis with the `--no-dest-tape-segregation` command line option. See the repack(7) man page for details.

HPSS_CORE_SEG_CACHE_IDLE_TIME

Adjusts the time, in seconds, a disk or tape storage segment can remain idle in the Core Server's in-memory cache before being purged. If this variable is not present in the environment, the server's default value is 60 seconds.

HPSS_CORE_TAPE_CACHE_IDLE_TIME

Adjusts the time, in seconds, a tape's in-memory cache entries can remain idle before being purged. This value controls the tape storage map, VV and PV caches. If this variable is not present in the environment, the server's default value is 300.

HPSS_CORE_LARGE_SEG_THRESHOLD

Controls the way tape storage space is allocated when large tape storage segments are written. The value of this environment variable is a time, measured in seconds. If the segment to be written would take longer to write than this time threshold, then the segment is written to a tape VV that has sufficient free space. This may mean the system will mount a different tape if it had been writing a tape that is nearly full. If the segment will take less time to write than this threshold, then the segment is written to any available tape, including a tape that will soon reach End Of Media (EOM). The server's default value is 30 seconds. The server uses the Transfer Rate information from the storage class when estimating the time required to write a segment.

This threshold value also applies when repack(7) is moving tape storage segments from one tape to another. Large tape storage segments will be moved to a tape that has sufficient free space if the time to move the segment exceeds the threshold value. Tapes selected for repack output are also subject to the HPSS_CORE_REPACK_OUTPUT limitation. When repack(7) creates new tape aggregates using small files, it writes to any available tape, subject to the rules defined by HPSS_CORE_REPACK_OUTPUT.

HPSS_SOCK_KEEPIDLE_SECS

Adjusts the time, in seconds, that a socket connected to a Mover will be idle before TCP pings the Mover. If the pings are not answered, TCP returns an error to the Core Server, after about 11 more minutes. The error allows the Core Server to recover control of a disk or tape transfer if the Mover crashes or if the node the Mover runs on crashes. Note that this timer does not help the Core Server regain control of a transfer if the Mover hangs up on a storage device. The default value of this setting is 7200 seconds (2 hours).

HPSS_CORE_DISKSEGCACHE_MEM_PERCENT

The Core Server's disk segment cache occupies, by default, 1% of total system memory. For example, on an HPSS Core Server machine with 256 GB of memory, the disk segment cache will use up to 2.56 GB of memory. Increasing this value beyond 1% may lead to increased file creation and deletion rates, at the cost of increased memory usage. Decreasing this value below 1% may lead to decreased file creation and deletion rates, with the benefit of decreased memory usage.

HPSS_CORE_DISCOVERY_THREADS

The number of threads dedicated to identifying boundaries and volumes for incoming request queue entries. This value impacts how quickly entries get placed in the queue for processing, and should be at least 10% of the `HPSS_CORE_REQUEST_THREADS`, but does not need to match `HPSS_CORE_REQUEST_THREADS` for larger values.

HPSS_CORE_REQUEST_THREADS

The number of threads dedicated to processing request queue entries. Ideally this would represent the desired maximum number of read requests, and should be at least the maximum number of tape drives available. Request threads will bind to a tape drive and process the requests present for that tape drive. Other free threads will process requests from disk in parallel.

HPSS_SS_SCHEDULE_BY_CONTROLLER

When this environment variable is set to any value, the Core Server will balance disk I/O scheduling by the disk device controller ID rather than the Mover ID. See Performance for more information.



Advice

_For more information on tape ordered recall, see Overview of TOR

16.2.2.2. Core Server I/O limit configuration

The number of concurrent IO operations allowed in each Core Server is influenced by several factors, including three parameters from the Core Server configuration. Two of these are in the **Core Server-specific configuration** tab of the *Core Server Configuration* window in SSM:

- Maximum Active I/O Reqs
- Maximum Active Copy Reqs

The other is in the **Interface Controls** tab:

- Thread Pool Size

The Core Server tracks the current number of IO requests in three categories:

- Active Total IO
- Active Copy IO
- Migrate IO

Active Copy IO includes stages, internal copies of one hpss file to another, and COS changes. Active Total IO includes all of the Active Copy IO plus reads by client apps from hpss and writes to hpss by client apps. Migrate IO includes only migrations.

Active Copy IO is a subset of Active Total IO. Migrate IO is in a class by itself; it is neither a subset nor a superset of Total IO nor of Copy IO.

The maximum values allowed for Active Total IO and Active Copy IO may be configured by changing the "Maximum Active I/O Requests" and "Maximum Active Copy Requests" parameters in the **Core Server-specific configuration**, respectively. There is no configurable maximum for Migrate IO.

IO requests submitted after these maximum values are reached will be rejected. However, up to 2000 additional stage requests can also be submitted to be run as background stages, regardless of the configured max values. These background stage requests are queued until there is room to run them, that is, until one or more of them can be handled without going over our maximum Active Copy IO limit. These background stage requests are not counted in the Active Copy IO until they are released from the queue and actually submitted for the stage.

The Core Server may adjust the actual values it uses for the Maximum Active Total IO and Maximum Active Copy IO based on the "Thread Pool Size" parameter in the *Core Server Configuration*. The Thread Pool Size specifies the total number of threads the Core Server may create. It must be large enough to allow for reasonable operation, and it must be large enough for the Maximum Active Total IO plus 20 threads reserved for Core Server internal use.

The bare minimum for "reasonable operation" is 25 threads. If Thread Pool Size is less than 25, the Core Server changes its actual Max Active Copy IO and Max Active Total IO to very small values, no greater than 10 each, and in some cases to 0 each.

If Thread Pool Size is greater than 25, but is still not large enough for the Maximum Active Total IO plus 20 threads reserved for Core Server internal use, then the Core Server reduces the actual max values it will use in this way:

1. It sets its actual Maximum Active Total IO to 20 less than the Thread Pool Size:
    - Actual Max Active Total IO = Thread Pool Size - 20
2. It calculates the percentage of Active Copy IO with regard to Active Total IO which was requested in the configuration:

```
percentage = (configured Max Active Copy IO) /
```

```
            (configured Max Active Total IO)
```

3. It sets its actual Maximum Active Copy IO to this percentage of its actual Maximum Active Total IO:

```
Actual Max Active Copy IO = percentage * Actual Max Active Total IO
```

So the site should configure Max Active Copy IO large enough to handle its expected peak concurrent number of stages, internal file copies, and COS changes.

It should configure Max Active Total IO large enough to handle all of the Active Copy IO plus enough for all the expected concurrent reads and writes.

It should configure Thread Pool Size large enough to handle Max Active Total IO, plus the number of migrations it expects to have, plus at least 20 for Core Server internal threads.

Changes to the Thread Pool Size take effect only after clicking **Apply Config** on the *HPSS Health and Status* screen or on a Core Server restart or reinitialization.

The SSM **Core Server-specific configuration** window shows the values of the Max Active Total IO and Max Active Copy IO as they existed in the **Core Server-specific configuration** based on the last time a configuration update was applied.

To see the adjusted values, look in the *Core Server Configuration* window or the **Core Server Information** window in SSM. Alternatively, use the hpsslimits(7) command, or issue a SIGHUP to the Core Server.

Here is an example of the hpsslimits command:

```
# hpsslimits
             Max Open Bitfiles =               18000
             Cur Open Bitfiles =                   0
                 Max Active IO =                 980
                 Cur Active IO =                   0
            Max Active Copy IO =                 607
            Cur Active Copy IO =                   0
               Max Connections =                2000
               Cur Connections =                   4
            Tape Queue Method =               Offset
         Tape Queue Active Set =                1024
          Allow Any Characters =                true
              Trashcan Enabled =               false
      Trashcan Unlink Eligible =              864000
                XML Size Limit =              131072
            XML Request Limit =              204800
       UDA Search Object Limit =                4096
          UDA Search XML Limit =              131072
      UDA Search XML/Obj Limit =                2048
```

If a SIGHUP is sent to the Core Server, examine the "BFS IO State" section of the resulting dump file. Here is an example of this section:

```
=============  BFS IO State  ===============
Total Number of Active IO Requests        210
Max Number of IO Requests Allowed         300
Total Number of Active Copy Requests      180
Max Number of Copy Requests Allowed       200
Total Number of Active Migrate Requests   40
```

Applications can retrieve the values for the configured limits via the API using the hpss_GetSubSysLimits function.

For read and stage queuing (such as stage callbacks), there are separate limits controlling how many threads are handling the queue.

Maximum RQ Request Threads

The number of threads dedicated to identifying boundaries and volumes for incoming request queue entries. This value impacts how quickly entries get placed in the queue for processing.

Maximum RQ Discovery Threads

The number of threads dedicated to processing request queue entries. This should be the number of tape drives desired to use for queued read and stage operations. Request threads will bind to a tape drive and process the requests present for that tape drive. Other free threads will process requests from disk in parallel.

16.2.2.3. Core Server max open files configuration

There are two settings used in determining the maximum number of concurrently open bitfiles allowed in HPSS: one setting for files opened by tape aggregation and one setting for files opened for any other purpose.

The values of the two settings are added together to reach the total number of files allowed to be open at any one time. In enforcing this limit, the Core Server does not check whether a file is being opened for tape aggregation or for some other reason. Having the two separate values is only a coarse means of helping the administrator estimate the needs of the system based on the system configuration and workload.

The limit for the max number of files opened for purposes other than tape aggregation is set in the **Core Server-specific configuration** in the "Maximum Open Bitfiles" field.

The limit for the max number of files opened concurrently for tape aggregation is hard-coded into the Core Server as 20,000; however, it may be changed by setting the environment variable HPSS_MAX_NUM_OPEN_FILES_FOR_AGGR to the desired value. The Core Server may use a value less than this limit based on its calculation of what is actually needed for aggregation.

On startup, the Core Server computes how many open files would be required for tape aggregation if an aggregate migration were executing concurrently on all possible combinations of storage classes and hierarchies. This depends upon several factors:

1. The number of disk storage classes.
2. The number of hierarchies that use each disk storage class.
3. The copy count in each affected hierarchy.
4. The "Total Migration Streams" defined in the migration policy of each storage class.
5. The number of batch migration requests allowed concurrently per stream, which is 2.
6. The max number of files allowed per aggregate, which is 50,000.

The calculation for the number of open files required for aggregate migration for one storage class as used in one hierarchy would be:

```
50,000 * stream count * 2 * copy count
```

The Core Server adds up these values for all disk storage classes as they are used in all hierarchies. The resulting value is not allowed to exceed HPSS_MAX_NUM_OPEN_FILES_FOR_AGGR, or, if the HPSS_MAX_NUM_OPEN_FILES_FOR_AGGR variable is not set, then it is not allowed to exceed 20,000.

This value is then added to the value of "Maximum Open Bitfiles" from the **Core Server-specific configuration** for the actual maximum open files allowed for the entire system.

To be recognized, changes in either value require a reinitialization or, sometimes, a restart of the Core Server. If the new total value would be less than the allocated size of the open context list, it can be implemented with a reinitialization. Changes to a larger value require a Core Server restart. The allocated size of the open context list is 4 times the initial value of the maximum number of open files.

A SIGHUP dump of the Core Server shows the current maximum open files allowed and the actual size of the open context list:

```
    Maximum open files  = 6000
    Allocated Size of Open Context List  = 24000
```

When the Core Server starts up, it issues a log message saying how many additional files it allowed for aggregation. The message looks something like this:

```
Tape aggregation on, adjusting open file cache: additional open files added=20000
```

If at startup the Core Server realizes it needs more open files for aggregation than allowed by the value of HPSS_MAX_NUM_OPEN_FILES_FOR_AGGR (or more than 20,000, if HPSS_MAX_NUM_OPEN_FILES_FOR_AGGR is not set), then it issues a warning message that looks something like this:

```
Tape aggregation on, adjusting open file cache:
additional open files needed=5700000,
```

```
limited by max = 20000 (Too many open bitfiles)
```

16.2.3. Gatekeeper-specific configuration

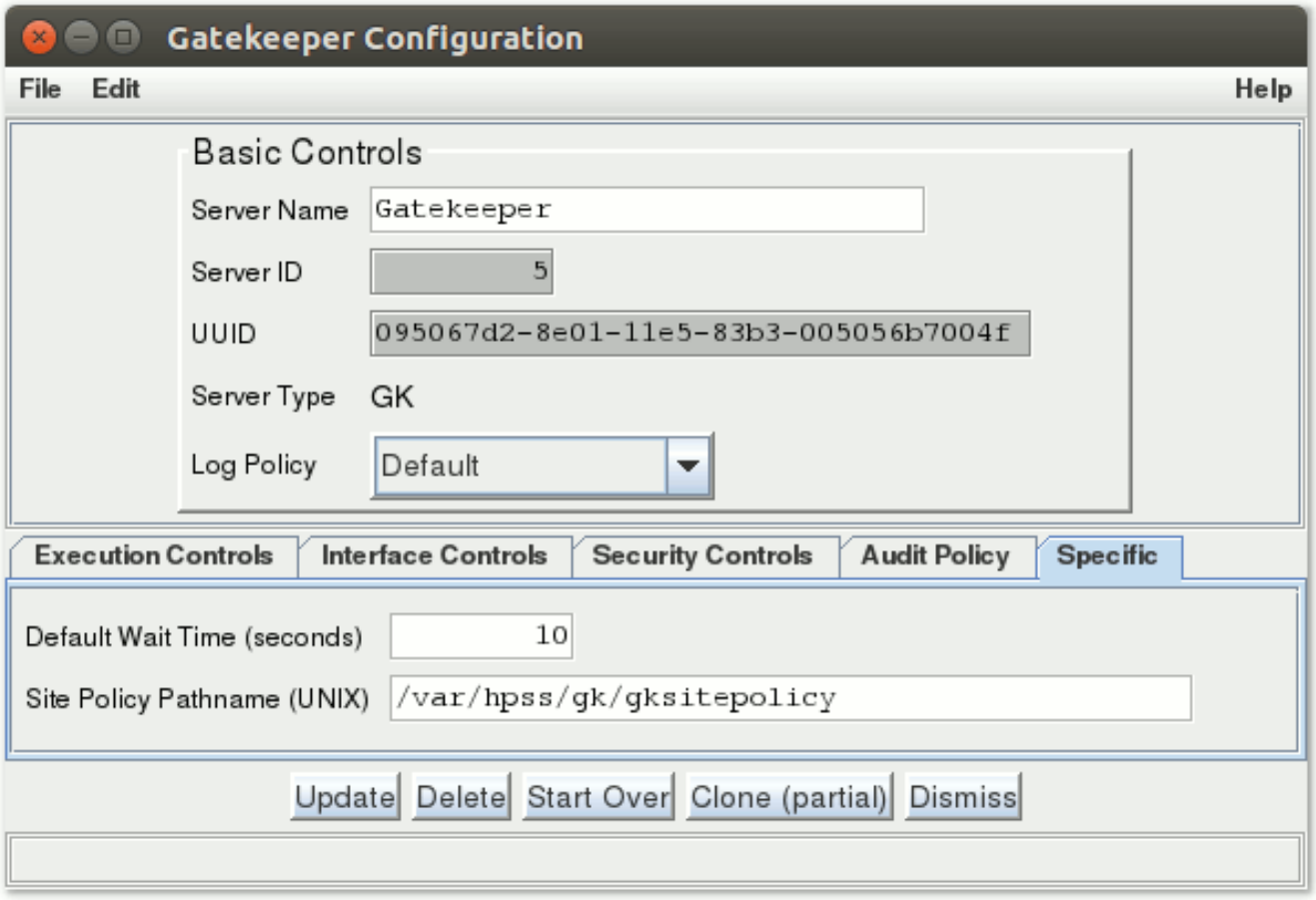
To use a Gatekeeper for gatekeeping services the Gatekeeper must be configured into one or more storage subsystems (see Storage Subsystem Configuration window). To associate a Gatekeeper with a storage subsystem, the Gatekeeper must be selected on the *Storage Subsystem Configuration* window.

To use the Gatekeeper for account validation services, account validation must be enabled in the accounting policy (see Accounting).

If account validation is configured to be ON, then at least one Gatekeeper must be configured even if the site does not wish to do gatekeeping. In this case, the Gatekeeper will only be used for validating accounts. Otherwise a Gatekeeper may be used for both gatekeeping and account validation. If multiple Gatekeepers are configured, then any Gatekeeper may be contacted for account validation requests.

Note: If a Gatekeeper is configured, then it will either need to be running or marked non-executable for HPSS Client API requests to succeed in the Core Server (even if no gatekeeping nor account validation is occurring); this is due to the HPSS Client API performing internal accounting initialization.

For a detailed description of the Gatekeeper, refer to the Gatekeeper section.



Field descriptions

Default Wait Time (seconds)

The number of seconds the client must wait before retrying a request. This value must be greater than zero and is used if the Gatekeeping Site Interface returns a wait time of zero for the create, open, or stage request being retried.

Site Policy Pathname (UNIX)

The contents of this file will be defined by the site and should be coordinated with the Gatekeeping Site Interface. For example, a site may want to define the types of requests being monitored, the maximum number of opens per user, or a path name to a log file. The **Site Policy Pathname** will be passed to the gk_site_Init routine upon Gatekeeper startup. If a site wishes to make use of the **Site Policy Pathname** file, then the site will need to add code to the Gatekeeping Site Interface library to read the **Site Policy Pathname** file and interpret the data accordingly. Refer to the *Site Interfaces* chapter of the *HPSS Programmer's Reference* for details on the site's interface library.

16.2.4. Core Server additional configuration

Some additional configuration outside SSM is required whenever a Core Server is added. Each client application which accesses HPSS must first contact the Core Server using the Core Server's RPC endpoints, which must be listed in the `/var/hpss/etc/ep.conf` file to be available to client applications. The **hpss_bld_ep** utility reads the Core Server configuration and creates the `ep.conf` file. The /var/hpss files section contains more information on the `ep.conf` file.

The **hpss_bld_ep** utility must only run on the HPSS root subsystem machine. To invoke the utility:

```
% /opt/hpss/config/hpss_bld_ep -v -f /var/hpss/etc/ep.conf -r <UNIX/LDAP
Realm Name> -c <schema name> -d <DB2 configuration database> add
```

For example:

```
% /opt/hpss/config/hpss_bld_ep -v -f /var/hpss/etc/ep.conf -r
hpss.acme.com -c hpss -d cfg add
```

The location service client-side library will cache location information locally on client systems. The cache is located in the HPSS application's home directory (~/.hpss_ls_cache) with permissions 0600 and is used to improve the startup time of client applications. This feature can be disabled by setting LS_DISABLE_LOCAL_CACHE=1 in the client system's HPSS `env.conf`.

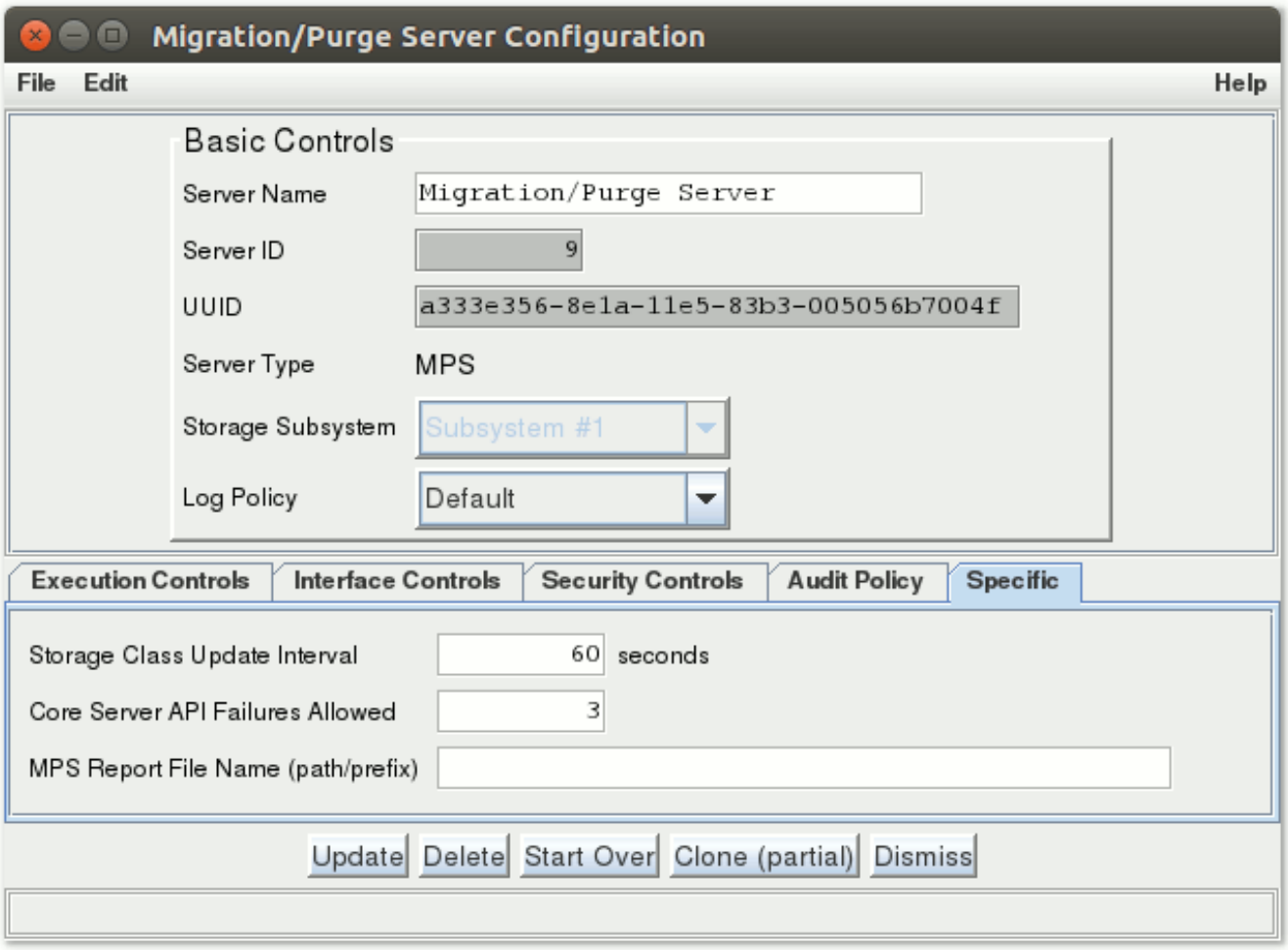
16.2.5. Core Server Behavior Under an Invalid License

The Core Server logs an increasing number and severity of alarm messages as an installed license gets nearer to being expired or has become expired but is within the grace period. It is important to contact HPSS support prior to the license expiring to retain full access to the features of HPSS.

When the Core Server detects an invalid or expired license, it logs a major alarm and will begin rejecting new write requests. Migrations, repacks, deletions, and other internal requests will not be impacted. Read requests out of the system will be limited to 5 concurrent requests.

Reach out to HPSS support and obtain a new license (see: Install HPSS License). The Core Server must be restarted after a valid license is applied.

16.2.6. Migration/Purge Server (MPS)-specific configuration



This tab of the *Migration/Purge Server Configuration* window allows you to update and view the type-specific configuration parameters for a Migration/Purge Server (MPS).



*The server, if already running, must be reinitialized or recycled in order for the changes to become effective. It is recommended to use the **Apply Config** button on the HPSS Health and Status window to reinitialize the server.*

Field descriptions

Storage Class Update Interval

The interval that indicates how often the MPS will query the Core Server in its subsystem to get the latest storage class statistics. This is also the interval the MPS uses to check whether it needs to initiate a purge operation on the storage class based on the associated purge policy. The valid range for this field is 10 - 600 seconds.

Core Server API Failures Allowed

The maximum number of consecutive Core Server API failures allowed before MPS aborts a migration or purge run. This applies to both disk and tape migration as well as disk purge. If this

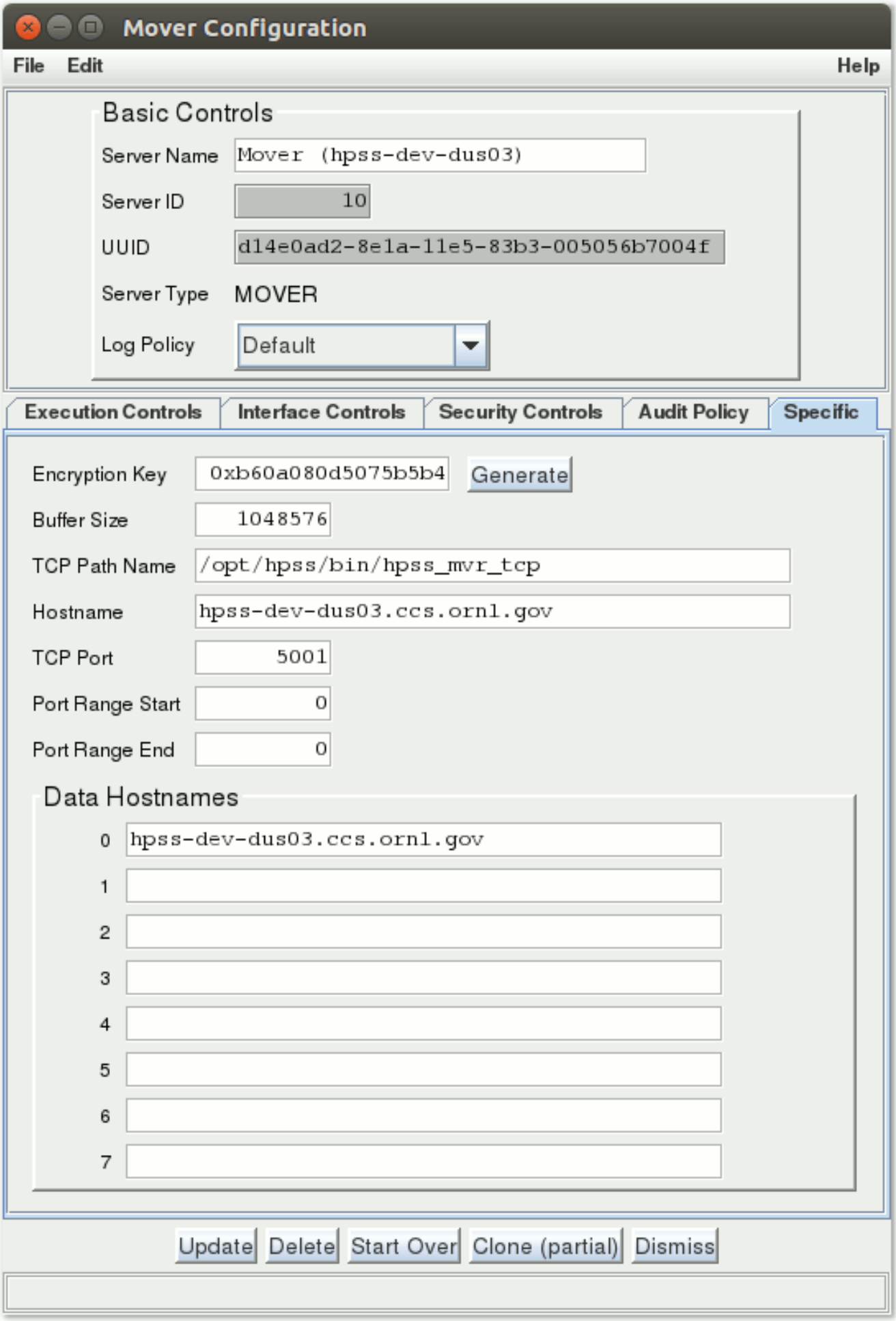
limit is reached during a disk migration run, migration skips to the next hierarchy to which that disk storage class belongs. Once the run has attempted to migrate all such hierarchies the run will end. If this limit is reached during a disk purge or tape migration run, the migration or purge run aborts.

MPS Report File Name (path/prefix)

A prefix string used by the MPS to construct a migration report file name. The full file name will consist of this string with a date string and subsystem ID appended to it. If this field is left blank, no migration reports will be generated. If a usable path prefix is not specified, the location of the migration report files may be unpredictable. A new MPS report file is started daily at midnight local time.

16.2.7. Mover-specific configuration

16.2.7.1. Mover-specific configuration window


Mover Configuration
File
Edit
Help
Basic Controls
Server Name
Mover (hpss-dev-dus03)
Server ID
10
UUID
d14e0ad2-8e1a-11e5-83b3-005056b7004f
Server Type
MOVER
Log Policy
Default
Execution Controls
Interface Controls
Security Controls
Audit Policy
Specific
Encryption Key
0xb60a080d5075b5b4
Generate
Buffer Size
1048576
TCP Path Name
/opt/hpss/bin/hpss_mvr_tcp
Hostname
hpss-dev-dus03.ccs.ornl.gov
TCP Port
5001
Port Range Start
0
Port Range End
0
Data Hostnames
0
hpss-dev-dus03.ccs.ornl.gov
1
2
3
4
5
6
7
Update
Delete
Start Over
Clone (partial)
Dismiss

This tab of the *Mover Configuration* window allows you to update and view the type-specific configuration for the Mover.

The server must be recycled in order for changes to become effective.

The Mover cannot be configured entirely from the SSM window. See Additional Mover configuration for further necessary configuration.

Field descriptions

Encryption Key

An encryption key used to secure the Mover's interface with the Core Server. A specific value may be entered in the field or a random value may be generated by clicking on the **Generate** button. The default value is "0". A nonzero value must be configured to allow client access to the Mover's data interface. See Additional Mover configuration for further information about the encryption key.

Buffer Size

The buffer size used for double buffering during data transfers. The default buffer size is set to 1,048,576 bytes (1 MB). This value should be tuned based on device and networking configuration and usage. The trade-off for this value is that large buffer sizes will use more system memory and may be inefficient for small transfers. For example, if the Mover buffer size is 4 MB, but client requests are 512 KB, the Mover will not achieve any double buffering benefit because the entire amount of the transfer fits in one Mover buffer. A smaller buffer size will cause device and network I/O to be interrupted more often, usually resulting in reduced throughput rates for all but the smallest transfers.

The minimum Mover buffer size is the smallest block size for any device the Mover will handle. For movers supporting object store devices, the Mover buffer size must be at least 64KB. The maximum value will be bounded by the available system memory and the number of concurrent Mover requests anticipated.

TCP Path Name

The pathname of the Mover TCP/IP listen executable. The default value is `/opt/hpss/bin/hpss_mvr_tcp`. The TCP Movers currently supported are listed below. All TCP Movers support common disk/tape interfaces, TCP/IP, and shared memory data transfers.

Table 20. Mover TCP pathname options

Name	Options supported
hpss_mvr_tcp	Standard disk/tape devices - tape devices that use the Native SCSI device driver

Hostname

The name of the host interface used for Mover control communication. This must be a valid hostname for one of the network interfaces on the Mover node. The **Execute Hostname** of the *Core Server Configuration* window is used as the default. If the Mover is running remotely, this field must correspond to a network interface on the remote node, whereas the **Execute Hostname** set in the Mover's basic configuration corresponds to a network interface on which

the administrative portion of the Mover runs.

Data Hostnames

The host network interface names to be used by the Mover when transferring data. There can be up to eight data hostnames specified. These must be valid hostnames for network interfaces on the Mover node. The default value of the first hostname is the **Execute Hostname** of the *Core Server Configuration* window. This field must correspond to a network interface on the remote node. These network interface addresses are utilized during migration/stage to allow the Mover to distribute connections and data across all the available network interfaces.

It is important to keep in mind that these hostnames are used by the Mover when it must supply network endpoints to be used to initiate connections from other Movers. These addresses are not used in the setup of connections between client and Mover (with the exception of the PData Push Protocol).

TCP Port

The TCP/IP port number used by the administrative interface portion of the Mover for the TCP/IP listen process to receive connections. It should be a value over 5000. The default value is 5001. The port number must be unique for processes running on the same node.

The Mover will internally use the port one greater than the one configured in this field on the remote node during initialization (for example, if 5001 is entered, port 5002 is used on the remote node); therefore available port ranges on both nodes must be taken into consideration when selecting this value.

Port Range Start

The beginning of a range of local TCP/IP port numbers to be used by the Mover when connecting to clients. Valid values are zero or any valid TCP port number to which the Mover may bind (that is less than or equal to the value of **Port Range End**). The default value is "0". Use of particular port ranges may be required by some sites for communication across a firewall. If this value is set to zero, the operating system will select the port number; otherwise the Mover selects a local port number between **Port Range Start** and **Port Range End** (inclusive). If nonzero, this field must be less than or equal to **Port Range End**. If this field is zero, **Port Range End** field must also be zero.

Port Range End

Used in conjunction with **Port Range Start** (see above). Valid values are zero or any TCP port number to which the Mover may bind (that is greater than or equal to the value of **Port Range Start**). The default value is "0". If nonzero, this field must be equal or greater than **Port Range Start**. If this field is zero, **Port Range Start** field must also be zero.

16.2.7.2. Additional Mover configuration

The part of the Mover that handles accessing configuration metadata and servicing the Mover administrative interface runs on the Core Server platform, while the part of the Mover (with which the PVL and Core Servers communicate) that manages the storage devices and performs data transfers runs on a remote node. To support this mode of operation, there is some additional configuration which must be done on the remote node.

This additional configuration may be performed using **mkhpss**. See Setting up remote Movers with **mkhpss**.

Alternately, the configuration may be performed manually using the instructions from the following sections:

/etc/services, /etc/inetd.conf, and /etc/xinetd.d

systemd

The Mover encryption key files

/var/hpss/etc files required for remote Mover

System configuration parameters on Linux

The Mover also responds to an environment variable, MVR_CLIENT_TIMEOUT. This environment variable may be set in `/var/hpss/etc/env.conf` on each remote Mover system. It represents the amount of time, in seconds, that the Mover will wait for messages from clients before timing out. The default timeout value of 60 minutes should be used in most circumstances. This should give plenty of time for an average tape drive to write the default maximum size aggregate.

If the max aggregate size is increased in the migration policy, this timeout may need to increase. The suggested value is about 133% of the time it takes to write that aggregate to tape. For example, a max aggregate size of 1TB with a transfer rate of 250MB/s should be set to 4000 (the duration of the transfer) * (4/3), for a resulting timeout of 5333 seconds.

Note that this timeout value affects Mover behavior in other ways, such as in error scenarios causing the Mover to require additional time to identify dropped clients and recover from it.

Similarly, the Mover responds to a MVR_PASSIVE_TIMEOUT environment variable. This environment variable may be set in `/var/hpss/etc/env.conf` and can be used to change the amount of time, in seconds, that a passive Mover process will wait before being contacted. This can be useful for stage operations which may have to wait a significant amount of time before being serviced. Consult with HPSS support before increasing this timeout value as it will impact Mover behavior in other ways, such as error and recovery scenarios.

/etc/services, /etc/inetd.conf, and /etc/xinetd.d

This is only relevant for systems which use **inetd** or **xinetd**. For HPSS systems running systemd, see systemd.

To invoke the remote part of the Mover, **inetd** on the remote node is utilized to start the parent process when a connection is made to a port based on the Mover's type specific configuration (see Mover-specific configuration).

An entry must be added to the `/etc/services` file so that **inetd** will listen on the port to which the Mover client process will connect. The port number is one greater than that configured for the **TCP Port** in the Mover's type-specific configuration. For example, if the configured listen port is 5001, the following line must be added to `/etc/services`:

```
hpss_mvr1 5002/tcp
```

For non-Linux systems

An entry must be added to the `/etc/inetd.conf` file which defines which executable to run and the arguments to use when a connection is detected. The entry must use the service name specified for the Mover in `/etc/services`. The sole argument (other than the program name) is the pathname to a file that contains an ASCII representation of the configured encryption key for the Mover (see the next section for further details). For example, for the `/etc/services` entry added above, the corresponding `/etc/inetd.conf` entry would be:

```
hpss_mvr1 stream tcp nowait root /opt/hpss/bin/hpss_mvr_tcp
hpss_mvr_tcp /var/hpss/etc/mvr_ek
```

This will cause **inetd** to run the executable `/opt/hpss/bin/hpss_mvr_tcp` under the root user ID when a connection is detected on port 5002. The Mover process uses the `/var/hpss/etc/mvr_ek` file to read the encryption key that will be used to authenticate all connections made to this Mover.

After modifying the `/etc/inetd.conf` file, be sure to refresh the inetd daemon using the following commands:

```
% ps -ef | grep inetd
root 6450 3154 0 Apr 29 - 0:02 /usr/sbin/inetd
hpss 17852 59370 2 16:50:25 pts/18 0:00 grep inetd
% kill -1 6450
```

For Linux systems

A file must be added to the `/etc/xinetd.d` directory which defines which program to run and the arguments to use when a connection is detected. The file will be given the same name as the service name specified for this Mover in `/etc/services`. For example, for the `/etc/services` entry added above, the corresponding file `/etc/xinetd.d/hpss_mvr1` would be created with the following contents:

```
service hpss_mvr1
{
    disable = no
    socket_type = stream
    protocol = tcp
    flags = IPv4
    wait = no
    port = 5002
    user = root
    server = /opt/hpss/bin/hpss_mvr_tcp
    server_args = -s /var/hpss/etc/mvr_ek
}
```

The specified port will be one greater than the port listed as the **TCP Port** in the Mover's type-specific configuration. For example, the port value in the example corresponds to a Mover with a **TCP Port** value of 5001.

This will cause **inetd** to run the executable `/opt/hpss/bin/hpss_mvr_tcp` under the root user ID when a connection is detected on port 5002. The Mover process will use the `/var/hpss/etc/mvr_ek` file to read the encryption key that will be used to authenticate all connections made to this Mover.

The **-s** option will enable syslog messages for startup debugging. This logging mechanism will be used prior to the initialization of the HPSS logging system and can be very useful in determining startup problems. The messages will be logged to the system log using the *user* facility and importance level of *alert*.

After modifying the file in `/etc/xinetd.d`, be sure to refresh xinetd daemon using the following commands:

```
% /sbin/service xinetd --full-restart
Stopping xinetd: [ OK ]
Starting xinetd: [ OK ]
```

systemd

This is only relevant for systems which use systemd. For HPSS systems running **xinetd**, see /etc/services, /etc/inetd.conf, and /etc/xinetd.d.

The following template files are provided for configuring the Mover with systemd:

```
/opt/hpss/config/templates/system_files/hpssmvr.socket.systemd.tmpl
/opt/hpss/config/templates/system_files/hpssmvr@.service.systemd.tmpl
```

The configured template files must be added to the `/etc/systemd/system/` directory. These can be copied into systemd as follows:

```
cp /opt/hpss/config/templates/system_files/hpssmvr.socket.systemd.tmpl \
/etc/systemd/system/hpssmvr.socket
cp /opt/hpss/config/templates/system_files/hpssmvr@.service.systemd.tmpl \
/etc/systemd/system/hpssmvr@.service
```

The file `/etc/systemd/system/hpssmvr.socket` would be created with the following contents:

```
[Unit]
 Description=HPSS Mover

[Socket]
 ListenStream=0.0.0.0:5012
 Accept=yes
```

```
[Install]
 WantedBy=sockets.target
```

The service file spawned by the socket has the following contents:

```
[Unit]
 Description=HPSS Mover Service
 Requires=hpssmvr.socket

[Service]
 Type=simple
 User=root
 ExecStart=/opt/hpss/bin/hpss_mvr_tcp -s /var/hpss/etc/mvr_ek
 StandardInput=socket
 StandardError=journal
 TimeoutStopSec=5

[Install]
 WantedBy=multi-user.target
```

The specified port will be one greater than the port listed as the **TCP Port** in the Mover's type-specific configuration. For example, the port value in the example corresponds to a Mover with a **TCP Port** value of 5011.

This will cause systemd to run the executable `/opt/hpss/bin/hpss_mvr_tcp` under the root user ID when a connection is detected on port 5012. The Mover process will use the `/var/hpss/etc/mvr_ek` file to read the encryption key that will be used to authenticate all connections made to this Mover.

The **-s** option will enable syslog messages for startup debugging. This logging mechanism will be used prior to the initialization of the HPSS logging system and can be very useful in determining startup problems. The messages will be logged to the system log using the *user* facility and importance level of *alert*.

After modifying the file in `/etc/systemd/system/`, be sure to refresh systemd using the following command:

```
% systemctl daemon-reload
```

Once the configuration files have been modified to have the desired settings, then the service can be enabled.

```
systemctl enable hpssmvr.socket
systemctl start hpssmvr.socket
```

Check that the systemd status of hpssmvr is active and running.

```
systemctl status hpssmvr.socket
```

The Mover encryption key files

To authenticate access to the remote Mover processes, the encryption key configured in this Mover's specific configuration (see Mover-specific configuration is read from a file accessible from the local file system. This file contains an ASCII representation of the encryption key. The pathname of the file is passed to the Mover executable as specified in the service file.

For example, if the encryption key in the Mover's type-specific configuration is `t7FdW/Hw4jOXRqS4Y0AjSW78vCUrCoUVq/Fybc0rw5o=`, then the encryption key file (`/var/hpss/etc/ek.mvr1`) should contain:

```
t7FdW/Hw4jOXRqS4Y0AjSW78vCUrCoUVq/Fybc0rw5o=
```

16.2.7.3. The Mover encryption cipher

The mover encryption cipher can be set in `/var/hpss/etc/site.conf`. To list available ciphers, run:

```
hpss_list_enc enc
```

If no cipher is specified, mover control communication will not be encrypted.

/var/hpss/etc files required for remote Mover

The Mover process on the remote machine requires access to the following files in `/var/hpss/etc`:

- `authz.conf`
- `env.conf`
- `HPSS.conf`

If the remote Mover process will have access to SAN3p devices, the following file is required.

- `hpss_san3p.conf`

The remaining files should be copied from the HPSS root subsystem machine.

In addition to the above files the Mover process on the remote machine requires that the `/var/hpss/tmp` directory be present with write and search permissions for the root user.

The /var/hpss files section contains a description of each of these files.

System configuration parameters on Linux

This section describes a number of system configuration parameters which may need to be modified for the successful operation of the remote Mover.

Refer to the Semaphore values section for detailed information on setting system parameters.

Setting up remote Movers with mkhpss

The **mkhpss** utility may be used to copy the files needed for a remote Mover from the root subsystem machine, to create the files which may not be copied, to install the required files on the remote Mover machine, and to configure the **inetd** to run the remote Mover.

To create the necessary files, run **mkhpss** on the root subsystem machine. From the **Root Subsystem Machine** menu, select the **Create Config Bundle** option. From this window, set the desired name of the config file bundle and click the **Create Config Bundle** button.

To install the files and configure the **inetd** on the remote Mover machine, transfer the created config bundle to the remote Mover machine and run **mkhpss** on the remote Mover machine. From the **Mover/Client Machine** menu, select the **Install Config Bundle** item. On this window, set the appropriate name of the bundle file and click the **Install Config Bundle** button.

Next, select the **Mover/Client Machine** menu **Configuration** submenu, and from this, the **Security Services** item. From this window, select the authentication mechanism (Kerberos or UNIX) using the drop down menu of the **Authentication Mechanism** field. For Kerberos, complete the remaining fields on the window and click the **Configure Security Services** button to set up the required files. For UNIX, there is nothing further to configure on this window.

Next, still under the **Mover/Client Machine** menu **Configuration** submenu, select the **Other Services** item. On this window, select the check box for **Configure any Movers for this machine**. This check box instructs **mkhpss** to add the proper lines for the Movers to `/etc/services`. It also instructs it, depending upon the operating system type, either to modify the `/etc/inetd.conf` file or to add the appropriate configuration file to the `/etc/xinetd.d` directory to support the Movers. On this same window, select the check box for **Set any system parameters necessary for HPSS**; this configures the local system settings described in System configuration parameters on Linux.

Mover configuration to support local file transfer

The Mover local file transfer (LFT) protocol allows the Movers to transfer data directly between a UNIX file system and HPSS.

Caveats:

- This transfer can only occur if both the Mover and the clients have direct access to the UNIX file system on which the file resides.
- If the multinode features of the Parallel FTP Client are used, the UNIX file system must be global to all nodes or unexpected results may occur.
- The Mover must be built with the LFT option. This is the default option for all Movers. If not all Movers have been built with this option, clients must explicitly specify a Class of Service which is valid for a Mover supporting the local file transfer option.
- The Mover must be running as root. If there are other Movers running on the same node, they must also run as root to take advantage of the Mover-to-Mover shared memory data transfer.
- A new configuration file (`/var/hpss/etc/hpss_mvr_localfilepath.conf`) is required on all nodes

running an enhanced Mover.

The configuration file contains a list of UNIX paths. These paths specify which files are allowed to be moved using this special data transfer protocol. Any UNIX file on the Mover node whose path starts with a path in the list of entries is allowed to be transferred using the special data protocol.

If the configuration file does not exist, then the SSM will issue a minor alarm when the Mover starts, and no UNIX files will be transferred with the special protocol.

Here is an example configuration file:

```
# This file contains Unix paths that can be accessed by the
# local Movers on this node. If the client wishes to make a
# local file transfer and the start of the client path matches
# any of the paths in this list then the transfer will proceed,
# otherwise the Mover will not transfer the file.
#
# The format of this file is simply a list of paths, one per
# line.
/gpfs
/local/globalfilesystem
```

In the above sample configuration, any file under the path `/gpfs` or the path `/local/globalfilesystem` can be transferred using the special data protocol subject to the caveats specified above.

Extreme care should be taken when using this feature. The Mover will move data from any file in its list. If the UNIX file system is not global to all Mover nodes, the request will fail.



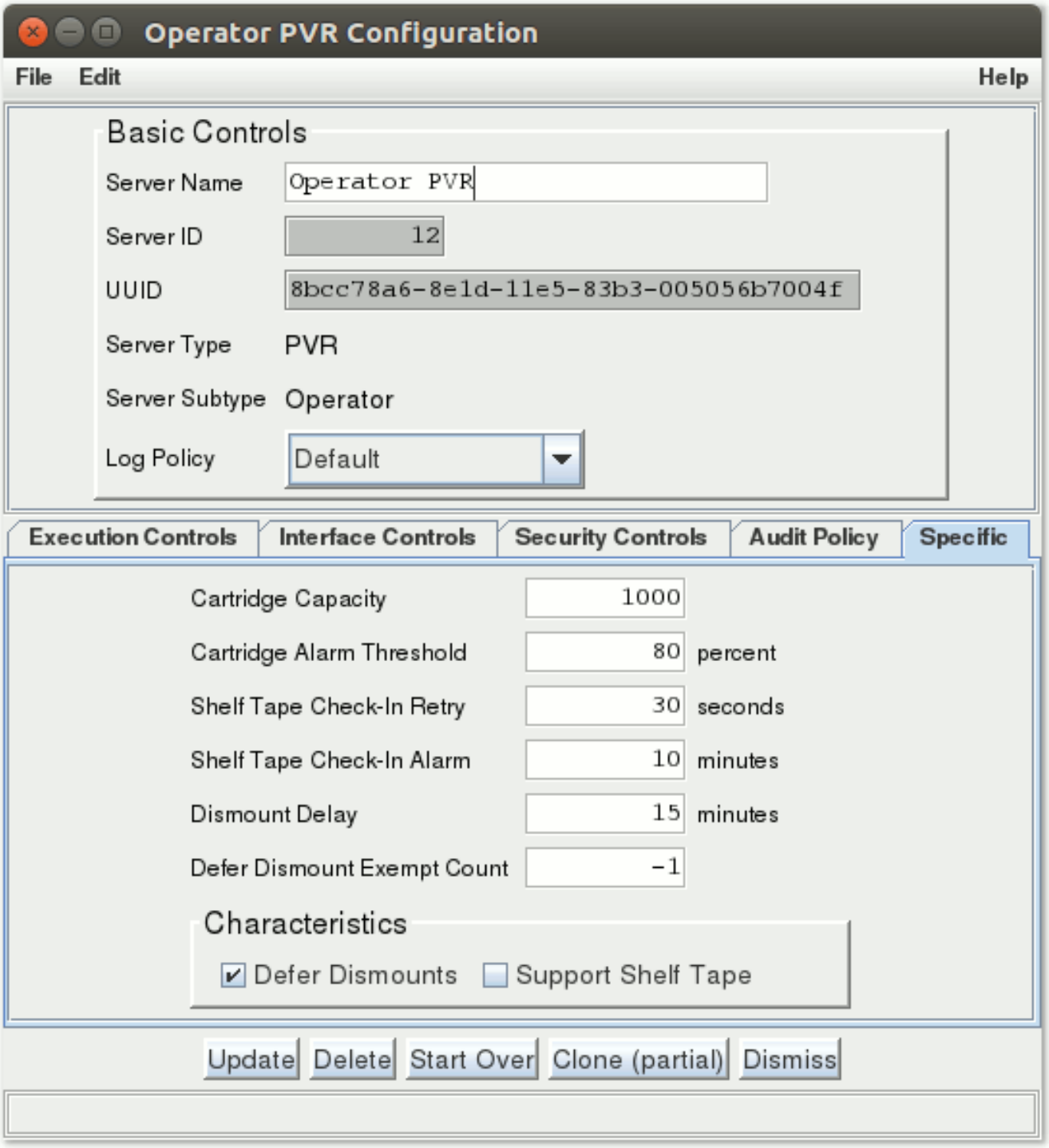
The following commands have been added to the HPSS Parallel FTP Clients to take advantage of these new features: lfget, lfput, mlfget, mlfput, lfappend. See the HPSS User's Guide *for more details. The hpss_ReadList and hpss_WriteList Client APIs can also be used to utilize the local file transfer capability. See the* HPSS Programmer's Reference - I/O Supplement *for more details.*

16.2.8. Physical Volume Repository (PVR)-specific configuration

The PVR-specific configuration entry can be created and managed using the *PVR Server Configuration* window. If changes are made while the server is running, the server must be recycled in order for the changes to become effective.

If you are configuring a PVR for StorageTek or generic SCSI: before proceeding with PVR configuration you should read the PVR-specific sections STK PVR-specific configuration window, SCSI PVR-specific configuration window. These sections provide additional vendor-specific advice on PVR/robot configuration.

16.2.8.1. Operator PVR-specific configuration window



Field descriptions

Cartridge Capacity

The total number of cartridge slots in the library dedicated to this HPSS PVR. This may or may not be the total cartridge capacity of the library; a site might use part of the library for some other HPSS PVR or for some non-HPSS application. The PVR uses the **Cartridge Capacity** field and the **Cartridge Alarm Threshold** field to determine when to send an alarm that the total cartridge threshold has been exceeded.

Cartridge Alarm Threshold

The percentage of the **Cartridge Capacity** at which the PVR will send an alarm.

Shelf Tape Check-In Retry

The number of seconds the PVR will wait before asking the robot if a requested shelf tape has been checked in. The PVR will continue checking at this interval until the tape is checked in. This field applies only if the **Support Shelf Tape** check box is selected. The retry value must be 30 or greater.

Shelf Tape Check-In Alarm

The PVR will periodically log alarm messages when a requested shelf tape has not been checked in. This field specifies the number of minutes between alarms. This field applies only if the **Support Shelf Tape** check box is selected. The alarm value must be "2" or greater.

Dismount Delay

When **Defer Dismounts** is checked, this value is used by the PVL to determine the number of minutes that dismounts are delayed after the last data access.

Defer Dismount Exempt Count

If set greater than "0", this value will specify the number of drives within a PVR that will be exempt from the defer dismount logic. This behavior will take place for all drives within the PVR with a drive/device "Library Unit ID" set (see Drive/Device Configuration). If multiple Library Unit IDs are set, the same Exempt count will apply to all Library Unit ID subsets.

The goal of this option is to allow a finer control of the defer dismount logic to account for transfer costs (for example, elevator, pass through costs). Defer dismount logic to apply logic on a sub library case (unit). The definition of a "unit" is placed in the hands of the administrator and likely based on library characteristics and topology.

Characteristics

Flags for the PVR:

Defer Dismounts

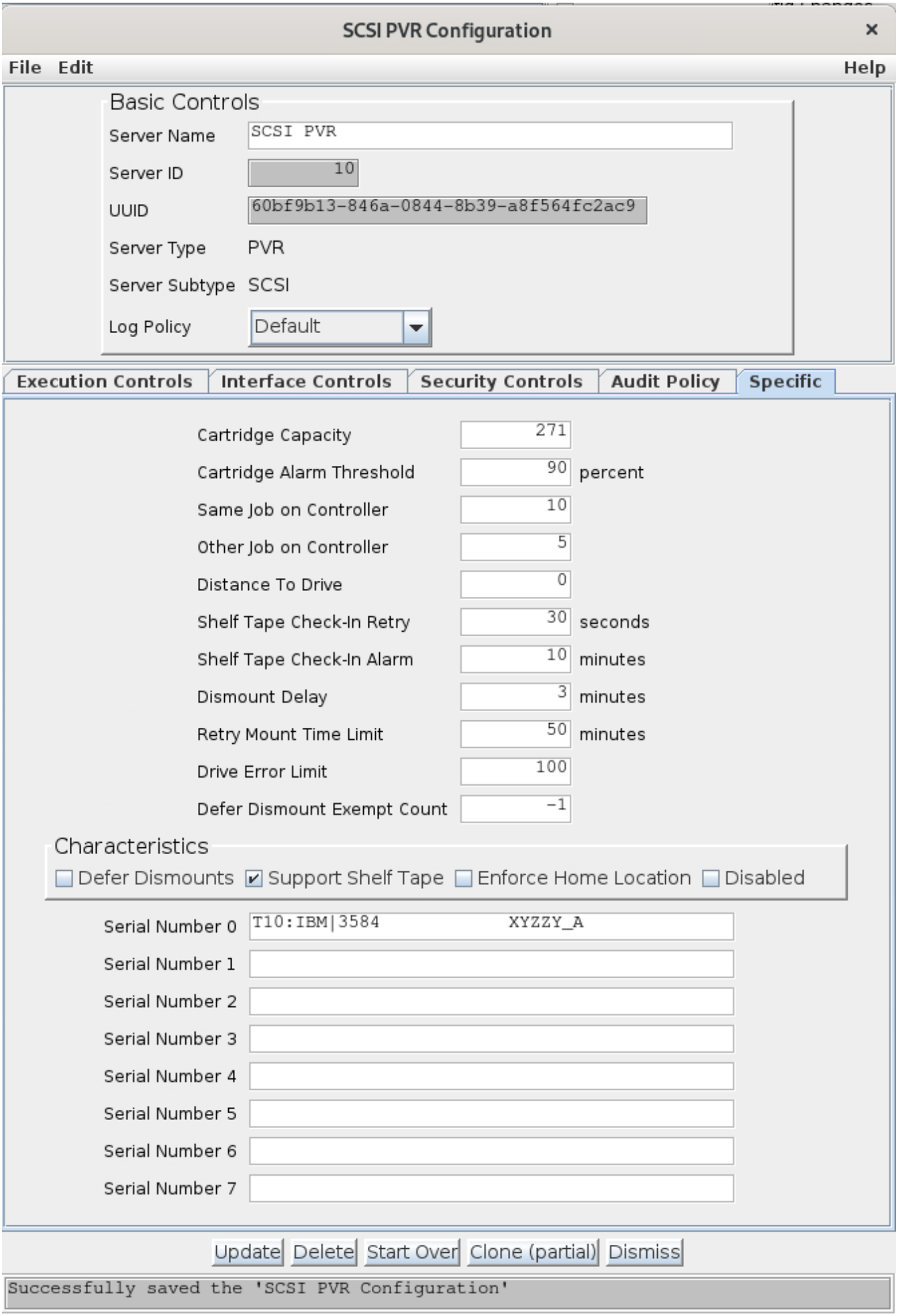
If ON, the PVL will delay the dismounting of a tape cartridge until the drive is required by another job or until the **Dismount Delay** time limit is exceeded. If the **Defer Dismount Exempt Count** is also set, the defer dismount logic will not be exercised on the number of drives set. This may result in free drives available for new mount requests without first generating defer dismount operations. See **Defer Dismount Exempt Count** for details.

A PVL restart is required in order for changes to this flag to take effect.

Support Shelf Tape

If ON, the PVR and the PVL will support the removal of cartridges from the tape library using the shelf_tape(7) utility.

16.2.8.2. SCSI PVR-specific configuration window


SCSI PVR Configuration
File Edit Help
Basic Controls
Server Name SCSI PVR
Server ID 10
UUID 60bf9b13-846a-0844-8b39-a8f564fc2ac9
Server Type PVR
Server Subtype SCSI
Log Policy Default
Execution Controls Interface Controls Security Controls Audit Policy Specific
Cartridge Capacity 271
Cartridge Alarm Threshold 90 percent
Same Job on Controller 10
Other Job on Controller 5
Distance To Drive 0
Shelf Tape Check-In Retry 30 seconds
Shelf Tape Check-In Alarm 10 minutes
Dismount Delay 3 minutes
Retry Mount Time Limit 50 minutes
Drive Error Limit 100
Defer Dismount Exempt Count -1
Characteristics
Defer Dismounts Support Shelf Tape Enforce Home Location Disabled
Serial Number 0 T10:IBM|3584 XYZZY_A
Serial Number 1
Serial Number 2
Serial Number 3
Serial Number 4
Serial Number 5
Serial Number 6
Serial Number 7
Update Delete Start Over Clone (partial) Dismiss
Successfully saved the 'SCSI PVR Configuration'

Cartridge Capacity

The total number of cartridge slots in the library dedicated to this HPSS PVR. This may or may not be the total cartridge capacity of the library; a site might use part of the library for some other HPSS PVR or for some non-HPSS application. The PVR uses the **Cartridge Capacity** field and the **Cartridge Alarm Threshold** field to determine when to send an alarm that the total cartridge threshold has been exceeded.

Cartridge Alarm Threshold

The percentage of the **Cartridge Capacity** at which the PVR will send an alarm.

Same Job on Controller, Other Job on Controller, Distance To Drive

These three values are used by the PVR when selecting a drive for a tape mount operation. The three values are essentially weights that are used to compute an overall score for each possible drive. After the score has been calculated, the drive with the lowest score is selected for the mount. If two or more drives tie for the lowest score, one drive is selected at random. The score is calculated as follows:

> Score =
> Weight 1 × Cartridges from this job mounted on this drive's controller +
> Weight 2 × Cartridges from other jobs mounted on this drive's controller +
> Weight 3 × Units of distance from the cartridge to the drive

This method has the effect of distributing a striped tape mount across as many controllers as possible for the best performance. It also will try to pick controllers that are currently driving a minimum number of tapes. So, in an environment with many tape drives per controller, the best performance will be achieved by minimizing the load on any one controller.

The **Distance To Drive** helps minimize mount times by mounting the tape in a physically close drive. All other things being equal, the tape will be mounted in the closest drive. However, if the closest drive is attached to a heavily used controller, then a more distant drive will be selected.

Shelf Tape Check-In Retry

The number of seconds the PVR will wait before asking the robot if a requested shelf tape has been checked in. The PVR will continue checking at this interval until the tape is checked in. This field applies only if the **Support Shelf Tape** check box is selected. The retry value must be "30" or greater.

Shelf Tape Check-In Alarm

The PVR will periodically log alarm messages when a requested shelf tape has not been checked in. This field specifies the number of minutes between alarms. This field applies only if the **Support Shelf Tape** check box is selected. The alarm value must be "2" or greater.

Dismount Delay

When **Defer Dismounts** is checked, this value is used by the PVL to determine the number of minutes that dismounts are delayed after the last data access.

Retry Mount Time Limit

The default value for this field is "-1". When the default value is used, if an error is encountered during a PVR mount operation, the mount will pend and be retried every five minutes. Setting a value in this field will change the mount behavior to periodically retry the mount until the specified time limit is exceeded. Once exceeded, an error is generated and the mount request is canceled. If the mount request would have resulted in a write operation, the error returned will cause the Core Server to set the **VV Condition** of the associated tape volume to DOWN. Once in DOWN state, the volume will no longer be available for read or write operations. For further information about the Core Server VV Condition, see Core Server Tape Volume information window.

Drive Error Limit

This field is intended to be used in conjunction with the PVR Server **Retry Mount Time Limit**. If the number of consecutive mount errors which occur to any drive in this PVR equal or exceed this value, the drive is automatically locked by the PVL. The only mount errors that apply are those set through the **Retry Mount Time Limit** mechanism. The **Drive Error Count** field in the *PVL Drive Information* records the number of consecutive errors on a drive by drive basis. To turn off the automatic drive disable feature, set the **Drive Error Limit** to "0" or "-1". Since this field is employed by the PVL, changing its value will not change drive disable behavior until the PVL is recycled.

Defer Dismount Exempt Count

If set greater than "0", this value will specify the number of drives within a PVR that will be exempt from the defer dismount logic. This behavior will take place for all drives within the PVR with a drive/device "Library Unit ID" set (see Drive/Device Configuration). If multiple Library Unit IDs are set, the same Exempt count will apply to all Library Unit ID subsets.

The goal of this option is to allow a finer control of the defer dismount logic to account for transfer costs (for example, elevator, pass through costs). Defer dismount logic to apply logic on a sub library case (unit). The definition of a "unit" is placed in the hands of the administrator and likely based on library characteristics and topology.

Characteristics

Flags for the PVR:

Defer Dismounts

If ON, the PVL will delay the dismounting of a tape cartridge until the drive is required by another job or until the **Dismount Delay** time limit is exceeded. If the **Defer Dismount Exempt Count** is also set, the defer dismount logic will not be exercised on the number of drives set. This may result in free drives available for new mount requests without first generating defer dismount operations. See **Defer Dismount Exempt Count** for details.

A PVL restart is required in order for changes to this flag to take effect.

Support Shelf Tape

If ON, the PVR and the PVL will support the removal of cartridges from the tape library using the shelf_tape(7) utility.

Enforce Home Location

If ON, the SCSI PVR will always try to dismount a mounted cart back to its home location. Otherwise, it will just use the first free slot. The scsi_home(7) utility can be used to view the home location values.

Disabled

If ON, the PVR is treated as disabled. This should be done when the underlying tape library is unusable. See Taking a tape library down for maintenance for more information on how this flag impacts operations.

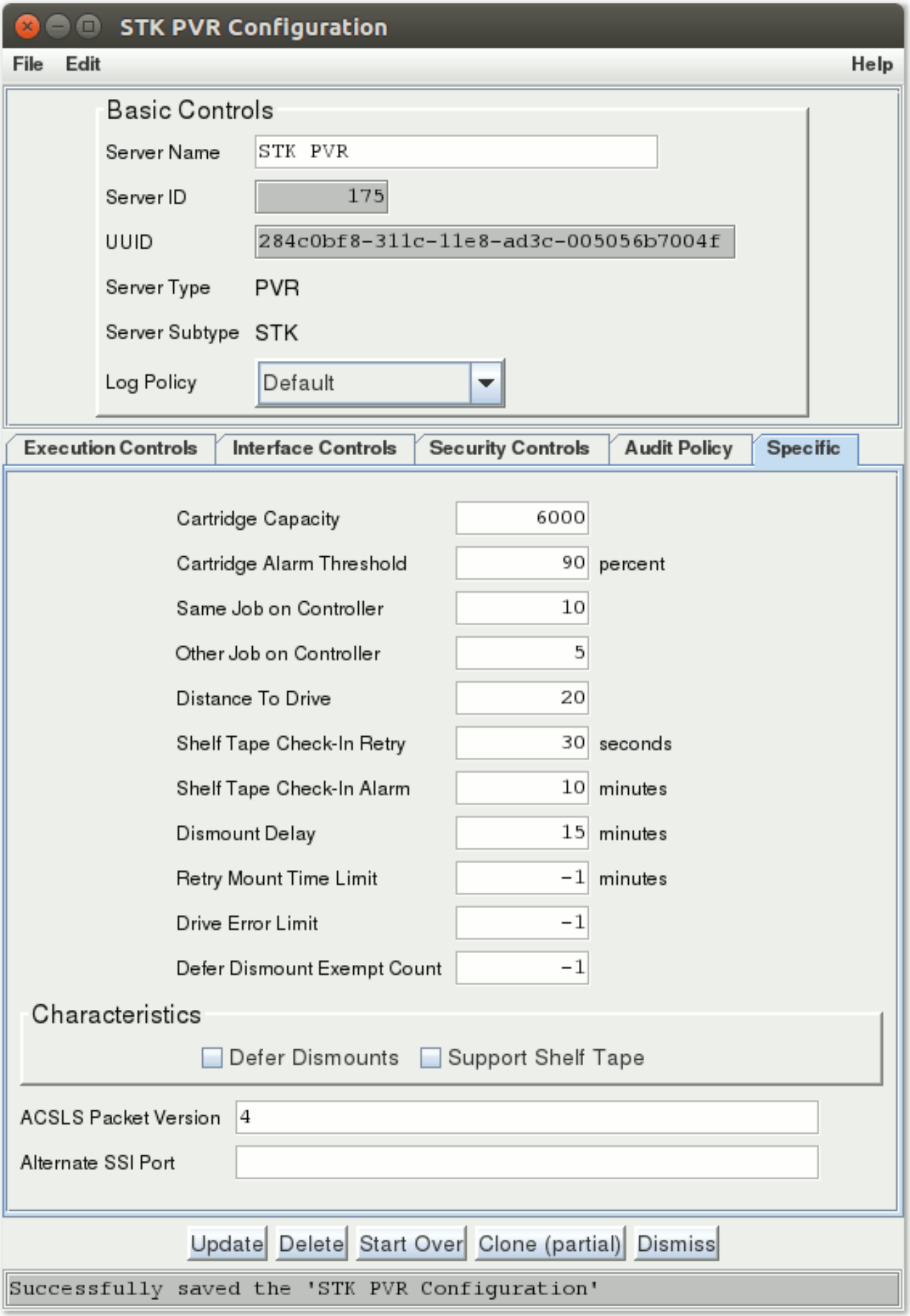
Serial Number 0 - 7

The serial number of the robots, obtained from device_scan(7). Only Serial Number 0 should be used. The serial number will allow the SCSI PVR to automatically look up all available control paths upon startup. Below is an example output from the device_scan(7) tool which shows the **library serial number** as well as the **drive serial number**. The **drive serial number** is used when configuring the library's drives and should be placed in the "Drive Address" field.

```
1) Device=[/dev/sg5]
    Vendor = [IBM]
    Product = [03584L22]
    Revision = [B07l]
    Vendor info = [B07l780000078A3467 0]
    Vital Product =
        Identifier = ['T10:IBM|03584L22        0000078A34670401'] // (Library serial
number)
        Data Transfer Elements
                    Address Serial
        1 = [257] ['T10:IBM|03592E06        0000078A426A'] // (Drive serial number)
        2 = [258] ['T10:IBM|03592E06        0000078A4276']
        3 = [259] ['T10:IBM|03592E06        0000078A4294']
        4 = [260] ['T10:IBM|03592E06        0000078A424D']
        5 = [261] ['T10:IBM|03592E06        0000078A4292']
        6 = [262] ['T10:IBM|03592E06        0000078A427A']
        7 = [263] ['T10:IBM|03592E06        0000078A4286']
        8 = [264] ['T10:IBM|03592E06        0000078A4261']
```

Note that everything between the single quotes should be included, including the internal spaces. The simplest way to do this is by copying and pasting the output into SSM.

16.2.8.3. STK PVR-specific configuration window



Field descriptions

Cartridge Capacity

The total number of cartridge slots in the library dedicated to this HPSS PVR. This may or may not be the total cartridge capacity of the library; a site might use part of the library for some other HPSS PVR or for some non-HPSS application. The PVR uses the **Cartridge Capacity** field and the **Cartridge Alarm Threshold** field to determine when to send an alarm that the total cartridge threshold has been exceeded.

Cartridge Alarm Threshold

The percentage of the **Cartridge Capacity** at which the PVR will send an alarm.

Same Job on Controller, Other Job on Controller, Distance To Drive

These three values are used by the PVR when selecting a drive for a tape mount operation. The three values are essentially weights that are used to compute an overall score for each possible drive. After the score has been calculated, the drive with the lowest score is selected for the mount. If two or more drives tie for the lowest score, one drive is selected at random. The score is calculated as follows:

> Score =
> Weight 1 × Cartridges from this job mounted on this drive's controller +
> Weight 2 × Cartridges from other jobs mounted on this drive's controller +
> Weight 3 × Units of distance from the cartridge to the drive

This method has the effect of distributing a striped tape mount across as many controllers as possible for the best performance. It also will try to pick controllers that are currently driving a minimum number of tapes. So, in an environment with many tape drives per controller, the best performance will be achieved by minimizing the load on any one controller.

The **Distance To Drive** helps minimize mount times by mounting the tape in a physically close drive. For STK drives, a unit of distance is from one Silo/LSM (Library Storage Module) to the next. This means that the tape must go through a pass-through port or elevator. This process is more time consuming. Therefore the default value for STK robots is a higher value which forces the tape to stay in the same Silo/LSM even if it means the drive selected is attached to a controller which is heavily used. All other things being equal, the tape will be mounted in the closest drive.

Shelf Tape Check-In Retry

The number of seconds the PVR will wait before asking the robot if a requested shelf tape has been checked in. The PVR will continue checking at this interval until the tape is checked in. This field applies only if the **Support Shelf Tape** check box is selected. The retry value must be "30" or greater.

Shelf Tape Check-In Alarm

The PVR will periodically log alarm messages when a requested shelf tape has not been checked in. This field specifies the number of minutes between alarms. This field applies only if the **Support Shelf Tape** check box is selected. The alarm value must be "2" or greater.

Dismount Delay

When **Defer Dismounts** is checked, this value is used by the PVL to determine the number of minutes that dismounts are delayed after the last data access.

Retry Mount Time Limit

The default value for this field is "-1". When the default value is used, if an error is encountered during a PVR mount operation, the mount will pend and be retried every five minutes. Setting a value in this field will change the mount behavior to periodically retry the mount until the specified time limit is exceeded. Once exceeded, an error is generated and the mount request is canceled. If the mount request would have resulted in a write operation, the error returned will cause the Core Server to set the **VV Condition** of the associated tape volume to DOWN. Once in DOWN state, the volume will no longer be available for read or write operations. For further information about the Core Server VV Condition, see Core Server Tape Volume information window.

Drive Error Limit

This field is intended to be used in conjunction with the PVR Server **Retry Mount Time Limit**. If the number of consecutive mount errors which occur to any drive in this PVR equal or exceed this value, the drive is automatically locked by the PVL. The only mount errors that apply are those set through the **Retry Mount Time Limit** mechanism. The **Drive Error Count** field in the *PVL Drive Information* records the number of consecutive errors on a drive by drive basis. To turn off the automatic drive disable feature, set the **Drive Error Limit** to "0" or "-1". Since this field is employed by the PVL, changing its value will not change drive disable behavior until the PVL is recycled.

Defer Dismount Exempt Count

If set greater than "0", this value will specify the number of drives within a PVR that will be exempt from the defer dismount logic. This behavior will take place for all drives within the PVR with a drive/device "Library Unit ID" set (see Drive/Device Configuration). If multiple Library Unit IDs are set, the same Exempt count will apply to all Library Unit ID subsets.

The goal of this option is to allow a finer control of the defer dismount logic to account for transfer costs (for example, elevator, pass through costs). Defer dismount logic to apply logic on a sub library case (unit). The definition of a "unit" is placed in the hands of the administrator and likely based on library characteristics and topology.

Characteristics

Flags for the PVR:

Defer Dismounts

If ON, the PVL will delay the dismounting of a tape cartridge until the drive is required by another job or until the **Dismount Delay** time limit is exceeded. If the **Defer Dismount Exempt Count** is also set, the defer dismount logic will not be exercised on the number of drives set. This may result in free drives available for new mount requests without first generating defer dismount operations. See **Defer Dismount Exempt Count** for details.

A PVL restart is required in order for changes to this flag to take effect.

Support Shelf Tape

If ON, the PVR and the PVL will support the removal of cartridges from the tape library using the shelf_tape(7) utility.

ACSLS Packet Version

The packet version used by STK's ACSLS software.

Alternate SSI Port

A site running two SSI (Server System Interface) services on the same platform will specify the non-default port number of the second service via this field.

STK PVR additional information

When configuring HPSS to manage an STK Robot, the Drive Address configuration entries correspond to the ACS, Unit, Panel, and Drive number used by ACSLS to identify drives. For example, the first drive in a typical STK Robot configuration has the Drive Address (0,0,10,0).

HPSS can share an STK Robot with other tape management systems. If a robot is shared, care must be taken to make sure that a drive is not used by any other tape management system while that drive is configured as unlocked in HPSS. This is important because HPSS can be configured to periodically poll all of its unlocked drives even if they are not currently mounted or in use by HPSS. If a drive is being used by another tape management system, it must be configured as locked in HPSS.

HPSS will use any Cartridge Access Port (CAP) in the STK Robot that has a priority greater than zero. When it needs a CAP, HPSS will pick the highest priority CAP that is currently available. At least one CAP must be assigned a nonzero priority. See the *STK Automated Cartridge System Library Software (ACSLS) System Administrator's Guide* for procedures to set CAP priority.

ACSLS Packet Version

See the *STK ACSLS System Administrator's Guide* for details. The environment variable ACSAPI_PACKET_VERSION will override the value entered in the **ACSLS Packet Version** field of the *STK PVR-specific configuration window*. If neither are set, a default value of "4" is used since ACSLS 7.x and 8.x generally use packet version 4.

Starting the STK client processes

In order for the HPSS PVR to communicate with the STK robot, it is required that STK client-side processes be running on the node where the HPSS PVR is executing. These client-side processes are the Server System Interface (SSI) and the Toolkit event logger. These binaries and associated script files are maintained by Oracle, but can be obtained from HPSS support.

The deliverable includes the binaries and script files necessary for starting the STK client-side processes as well as documentation for these files. Binaries are available for the Linux x86_64, and Linux PPC platforms.

The `t_startit.sh` file must be executed to start the STK client processes on those nodes where the HPSS PVR is executing. The script will prompt for the following options:

1. Multi-Host or Single-Host: specify "m" for multiple host.
2. Server side or Client side: specify "c" for client-side.
3. Remote Host Name: specify the name of the host where the ACSLS software is running.
4. Real ACSLS server or simulated test: specify "r" for real ACSLS server.
5. Remote Host Version: specify the remote host version which is listed for the ACSLS Release Number. This will usually be "4":

```
ACSLS Release Number        Remote Host Version Number
 5.x, 6.x, 7.x, 8.x                    4
```

6. Whether processes created should be listed (Y or N): specify either "y" or "n".
7. Whether t_cdriver should be started: specify "n".

To terminate the STK client processes, the script `t_killit.sh` may be executed.

Sample output from `t_startit.sh` follows:

```
             ***** Welcome to t_startit.sh, Version 2.3 *****
    This is the automated tester and startit script for the TOOLKIT.*
    Simply answer the questions which follow.*
    Executables live in: /opt/hpss/stk/bin*
Would you like Multi-Host or Single-Host testing?*
    Enter one of the following followed by ENTER:*
    M Multi-host testing*
    S Single-host testing*
    X eXit this script*
Enter choice: m*
Would you like to define the server side or client side*
for Multi-Host testing?*
    Enter one of the following followed by ENTER:*
    S Server side*
    C Client side*
Enter choice: c*
The Remote Host Name is the name of the server which*
has the ACSLS software (or simulator) running on it.*
Enter Remote Host Name (CSI_HOSTNAME): jeep*
Is that host running a real ACSLS (or LibStation) server,*
or a Simulated Test server (t_acslm)?*
     Enter one of the following followed by ENTER:*
     R Real ACSLS or LibStation Server*
     S Simulated Test Server*
Enter choice: r*
  The Remote Host Version number is the ACSLS Packet*
  Version level which the server is expecting.*
  Here are the valid choices:*
  ACSLS Release Number     Remote Host Version Number*
```

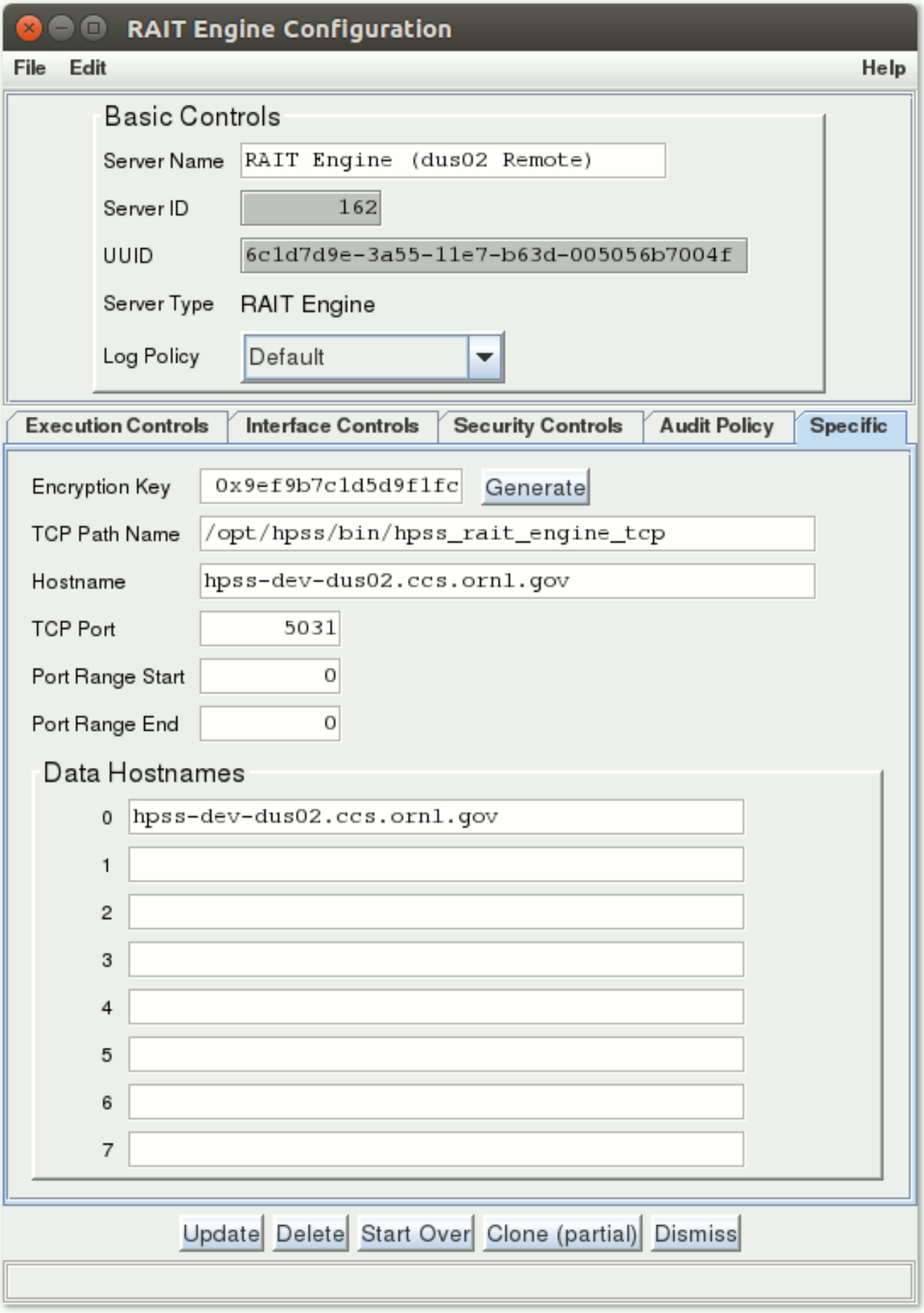
```
          3.x                          2*
          4.x                          3*
          5.x, 6.x, 7.x, 8.x           4*
Enter Remote Host Version (ACSAPI_PACKET_VERSION): 4*
Starting /opt/hpss/stk/bin/mini_el...*
Attempting startup of /opt/hpss/bin/mini_el ...*
Starting /opt/hpss/bin/ssi...*
Attempting startup of PARENT for /opt/hpss/bin/ssi...*
SIGHUP received Parent Process ID is: 17290*
Attempting startup of /opt/hpss/bin/ssi...*
SIGHUP received Parent Process #17290 EXITING NORMALLY*
Initialization Done.*
Do you want to see the processes created? (Y or N): y*
17288 p4 S 0:00 /opt/hpss/stk/bin/mini_el*
17292 p4 S 0:00 /opt/hpss/stk/bin/ssi 17290 50004 23*
17295 p4 S 0:00 grep /opt/hpss/stk/bin*
Do you want to start up t_cdriver? (Y or N): n*
```

IPv6 limitation

The HPSS STK PVR is not supported on IPv6 due to the Oracle StorageTek CDK ACSAPI library not supporting it at this time. If you require this support, consult your Oracle representative about implementing this enhancement.

16.2.9. RAIT Engine-specific configuration

16.2.9.1. RAIT Engine-specific configuration window



This tab of the *RAIT Engine Configuration* window allows you to update and view the type-specific configuration for the RAIT Engine.

The server must be recycled in order for changes to become effective.

Field descriptions

Encryption Key

An encryption key used to secure the RAIT Engine's interface with the Core Server. A specific value may be entered in the field or a random value may be generated by clicking on the **Generate** button. The default value is "0". A nonzero value must be configured to allow client access to the RAIT engine's data interface.

TCP Path Name

The pathname of the RAIT Engine TCP/IP process. The default value is "`/opt/hpss/bin/hpss_rait_engine_tcp`".

Hostname

The name of the host interface used for RAIT Engine control communication. This must be a valid hostname for one of the network interfaces on the RAIT Engine node. The **Execute Hostname** of the *Core Server Configuration* window is used as the default. If the RAIT Engine is running remotely, this field must correspond to a network interface on the remote node, whereas the **Execute Hostname** set in the RAIT Engine's basic configuration corresponds to a network interface on which the administrative portion of the RAIT Engine runs.

Data Hostnames

The host network interface names to be used by the RAIT Engine when transferring data. There can be up to eight data host names specified. These must be valid hostnames for network interfaces on the RAIT Engine node. The default value of the first hostname is the **Execute Hostname** of the *Core Server Configuration* window. This field must correspond to a network interface on the remote node. These network interface addresses are utilized during migration/stage to allow the RAIT Engine to distribute connections and data across all the available network interfaces.

TCP Port

The TCP/IP port number used by the administrative interface portion of the RAIT Engine for the TCP/IP listen process to receive connections. It should be a value over 6000. The default value is 6001. The port number must be unique for processes running on the same node.

The RAIT Engine will use a port one greater than the value in this field on the remote node during initialization (for example, if 6001 is entered, port 6002 is used on the remote node); therefore available port ranges on both nodes must be taken into consideration when selecting this value.

Port Range Start

The beginning of a range of local TCP/IP port numbers to be used by the RAIT Engine when connecting to clients. Valid values are zero or any valid TCP port number to which the RAIT Engine may bind, that is less than or equal to the value of **Port Range End**. The default value is "0". Use of particular port ranges may be required by some sites for communication across a firewall. If this value is set to zero, the operating system will select the port number; otherwise the RAIT Engine selects a local port number between **Port Range Start** and **Port Range End**

inclusive. If nonzero, this field must be less than or equal to **Port Range End**. If this field is zero, **Port Range End** field must also be zero. Note that in order for port range configuration to have an effect, client processes have to have the client API variable HPSS_API_USE_PORT_RANGE set (HPSS_API_USE_PORT_RANGE=1).

Port Range End

Used in conjunction with **Port Range Start** (see above). Valid values are zero or any TCP port number to which the RAIT Engine may bind (that is greater than or equal to the value of **Port Range Start**). The default value is "0". If nonzero, this field must be equal or greater than **Port Range Start**. If this field is zero, **Port Range Start** field must also be zero.

Configuring RAIT with systemd

Starting with RHEL 9 xinetd functional has been removed. Additional configuration is required to use systemd in its place. The following template files are provided for configuring RAIT with systemd:

```
/opt/hpss/config/templates/system_files/hpssrait.socket.systemd.tmpl
/opt/hpss/config/templates/system_files/hpssrait@.service.systemd.tmpl
```

The configured template files must be added to the /etc/systemd/system/ directory. These can be copied into systemd as follows:

```
cp /opt/hpss/config/templates/system_files/hpssrait.socket.systemd.tmpl \
/etc/systemd/system/hpssrait.socket
cp /opt/hpss/config/templates/system_files/hpssrait@.service.systemd.tmpl \
/etc/systemd/system/hpssrait@.service
```

Once the configuration files have been modified to have the desired settings, then the enable the socket.

```
systemctl enable hpssrait.socket
systemctl start hpssrait.socket
```

Check that the socket is enable and active

```
systemctl status hpssrait.socket
```

16.2.10. Deleting a server configuration

A server's configuration can be removed when the server is no longer needed. However, care should be taken to ensure that all objects created by the server are deleted properly and that all references to the server are removed from the HPSS configuration.



The steps described in this section are general guidelines. Specific procedures should

> *be worked out with the aid of HPSS support so that the details of the system's configuration can be considered.*

A server's configuration should be removed only when it is no longer needed. To modify a server's configuration, update the existing configuration instead of deleting the configuration and creating a new one. This will reduce the possibility of introducing configuration errors into HPSS.

There must be at least one Core Server, PVL, and Mover configuration entry in order for HPSS to be operational.

It is very important that the server's configuration entries be deleted in the order shown below.

To delete a server, perform the following steps in the order shown:

1. Delete all objects created by the server.
   - If deleting a Core Server configuration, realize that this should only be done when deleting the full subsystem.
   - If deleting a PVL configuration, all imported cartridges must first be exported.
   - If deleting a PVR configuration, all injected cartridges must either be exported or moved to another PVR.
   - If deleting a Gatekeeper, remove all references to that Gatekeeper from every subsystem configuration.
2. Remove references to the server from other configurations.

   Do not delete Root Core Server configuration.

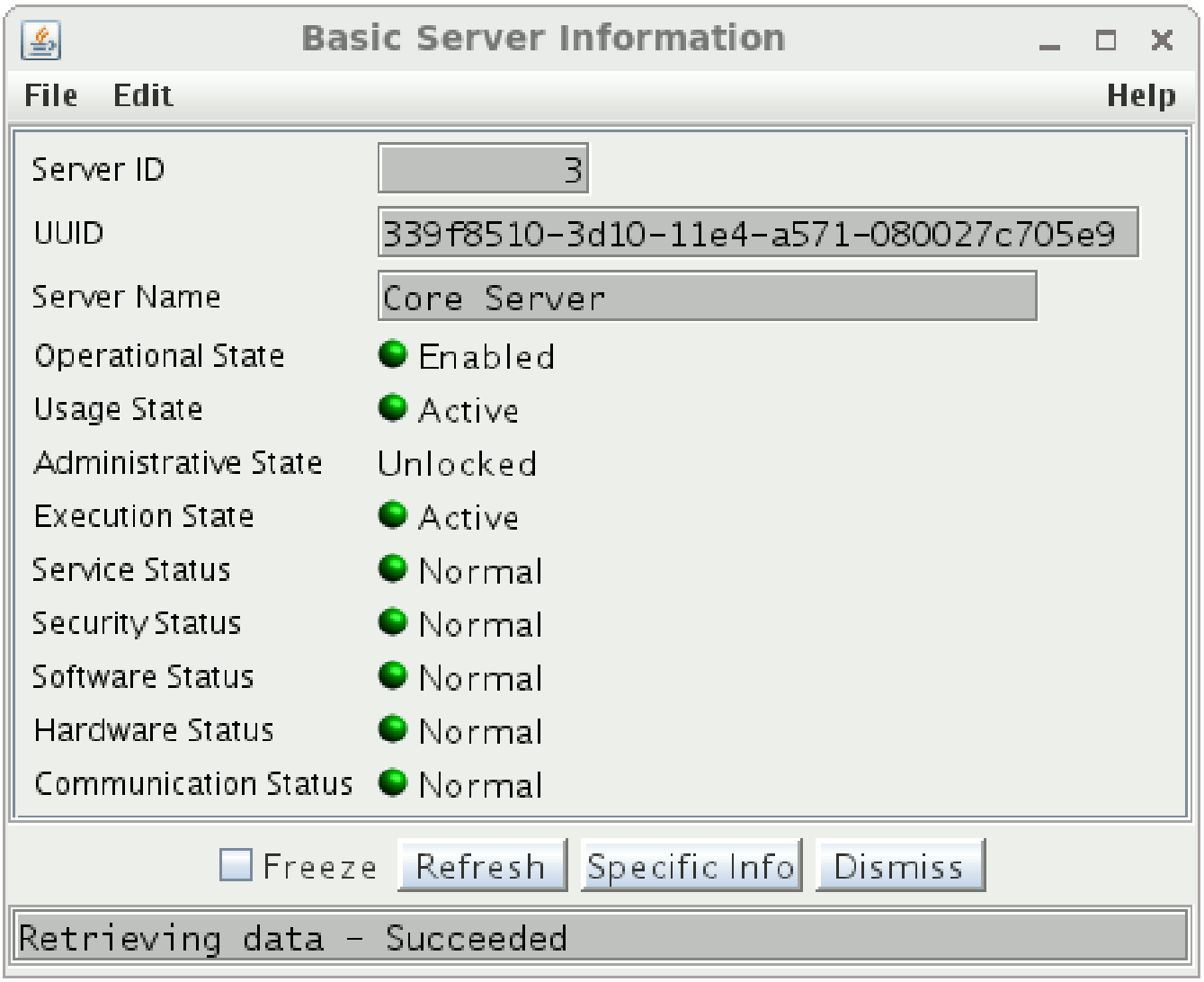
   - If deleting a PVR configuration, update the **PVR** field in the appropriate device and drive configuration using Devices and Drives window.
   - If deleting a Mover configuration, update the Mover field in the appropriate device and drive configuration using Devices and Drives window.
3. After the affected configurations are modified, reinitialize (if supported) or recycle the appropriate HPSS servers.
4. Ensure that the server is not running.
5. Delete UNIX files created or used by the server, if no longer needed.
   - If deleting the MPS, delete the MPS reports.
   - If deleting the Startup Daemon, delete all server lock files in the `/var/hpss/tmp` directory. The names of these files always begin with `hpssd`.
   - If deleting the Mover, delete Mover's socket files in `/var/hpss/tmp` directory.
6. Delete the server configuration by opening the server list, highlighting the target server, and clicking the **Delete** button.

16.3. Monitoring server information

A server that is running and connected to SSM will allow the SSM user to view and update its information. This section describes the server execution statuses and configuration information.

A typical HPSS server allows the SSM users to control its execution and monitor its server-related data through the *Basic Server Information* window and the server-specific information windows. These windows are described in the following subsections.

16.3.1. Basic Server Information



This window allows you to view the basic information associated with an HPSS server. The information is acquired from the server itself. While this window remains open, it will be automatically updated when the status of the server changes.

The *Basic Server Information* window is not be available for viewing if the server's operational state is `Broken`, `Not Enabled`, or `Unknown`, or the System Manager is unable to contact the server.

Field descriptions

Server ID

A unique value automatically generated and displayed for servers.

UUID

The universal unique identifier for the server.

Server Name

The descriptive name of the server.

Operational State

The operational state of the server. This state indicates the overall health of the server. Possible values are:

- `Enabled` indicates the server is running normally.
- `Disabled` indicates the server is not running because it has been shut down by SSM. You will not normally see this state on this window since the server must be running to provide the state information.
- `Suspect` indicates the server has seen an error that may indicate a problem is developing.
- `Minor` indicates the server has seen a problem which does not seriously impact HPSS operation.
- `Major` indicates the server has seen a problem which degrades HPSS operation.
- `Broken` indicates the server has shut itself down due to a serious error.

Usage State

The usage state of the server. Possible values are:

- `Active` indicates the server is working normally.
- `Idle` indicates the server is running, but has no work to perform. Most servers do not update usage state dynamically, so it is unlikely you will see this value reported.
- `Busy` indicates the server is busy performing its function. Most servers do not update usage state dynamically, so it is unlikely you will see this value reported.
- `Unknown` indicates the server has not reported a recognized usage state.

Administrative State

The Administrative State of the server. The possible states are:

- `Shut Down` indicates the server shut itself down in an orderly way.
- `Force Halt` indicates the server accepted a Force Halt command and terminated immediately.
- `Reinitialize` indicates the server reinitialized itself. Not all HPSS servers support being reinitialized.
- `Mark Repaired` indicates the server cleared its error states.
- `Unlocked` is displayed during normal server operation.

Execution State

The execution state of the server. Possible values are: `Active`, `Initializing`, `Restarting`,

`Terminating`, `Inactive`, `Suspending`, `Suspended`, and `Resuming`.

Service Status

The server's service status. Possible values are: `Normal`, `Warning`, `Minor Problem`, `Major Problem`, and `Critical`.

Security Status

The server's security status. Possible values are: `Normal`, `Warning`, `Minor Problem`, `Major Problem`, and `Critical`.

Software Status

The server's software status. Possible values are: `Normal`, `Warning`, `Minor Problem`, `Major Problem`, and `Critical`.

Hardware Status

The server's hardware status. Possible values are: `Normal`, `Warning`, `Minor Problem`, `Major Problem`, and `Critical`.

Communication Status

The server's communication status. Possible values are: `Normal`, `Warning`, `Minor Problem`, `Major Problem`, and `Critical`.

Using the Basic Server Information window

Most HPSS servers allow the SSM user to view the server's states and statuses.

Under *normal* conditions, the server states and statuses reported on the *Basic Server Information* window are as follows:

Operational State: `Enabled`

Usage State: `Active`

Administrative State: `Unlocked`

Execution State: `Active`

Service Status: `Normal`

Security Status: `Normal`

Software Status: `Normal`

Hardware Status: `Normal`

Communication Status: `Normal`

However, when the server is experiencing errors or encountering abnormal conditions, it will change the appropriate states and statuses to error values, notify SSM of the changes, and issue an alarm to SSM. Refer to Alarm/Event Information for more information.



The Startup Daemon and the System Manager do not have their own Basic Server Information windows.

16.3.2. Specific server information

Using the server-specific information windows

The majority of the HPSS servers also allow the SSM user to view and change their server-specific data through the SSM windows. A typical server allows authorized users to change the value of the fields and use them immediately in the current execution.

There are server-specific information windows for the following servers:

- Core Server
- Gatekeeper
- Migration/Purge Server
- Mover
- Physical Volume Library
- Physical Volume Repository
- RAIT Engine

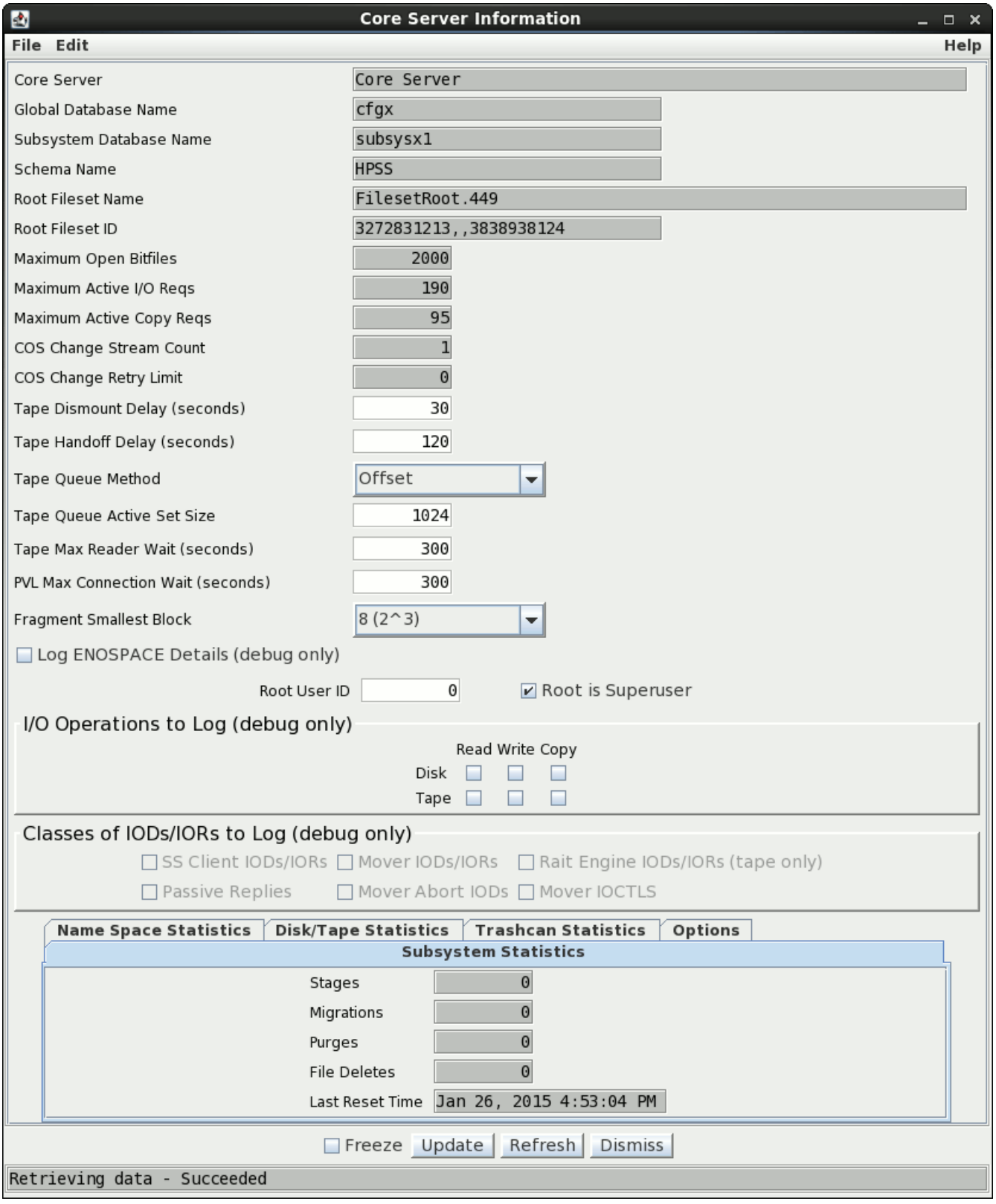


*Any changes made on a server's information window will be effective only during the server's current execution. To make the changes permanent, make the identical changes on the server's configuration window and if the setting changed allows, the change should be applied by clicking the **Apply Config** button on the HPSS Health and Status screen. Refer to Server reinitialization behavior to determine what settings can be applied by **Apply Config** (i.e reinitialization). Otherwise, restart the server (see Server configuration).*

If the server's operational state is `Broken` or `Unknown`, the specific server information window will not be available for viewing.

The server-specific information windows are described in detail below.

16.3.2.1. Core Server Information window



Field descriptions

Core Server

The descriptive name of the Core Server.

Global Database Name

The name of the global database for the HPSS system.

Subsystem Database Name

The name of the database which contains the subsystem tables used by this Core Server.

Schema Name

The name of the database schema.

Root Fileset Name

The name of the root fileset used by the Core Server.

Root Fileset ID

The fileset id for the root fileset used by the Core Server.

Maximum Open Bitfiles

The maximum number of bitfiles that can be open simultaneously.

Maximum Active I/O Reqs

The maximum number of simultaneous I/O requests allowed. See Core Server I/O limit configuration for a detailed description of how this field interacts with other configuration fields.

Maximum Active Copy Reqs

The maximum number of simultaneous copy requests allowed. See Core Server I/O limit configuration for a detailed description of how this field interacts with other configuration fields.

COS Change Stream Count

Indicates the number of background COS change threads to run in the Core Server. The default is "1".

COS Change Retry Limit

Limits the number of times that the Core Server will retry a failed COS change operation. When this limit is reached, the attempt to change the COS of the file is terminated.

Tape Dismount Delay (seconds)

The amount of time, in seconds, a mounted tape volume will remain idle before being dismounted by the Core Server. Larger values may reduce undesirable tape dismount/remount events, at the expense of lower tape drive utilization.

Tape Handoff Delay (seconds)

The amount of time, in seconds, a mounted tape will be held in a client's session before become eligible to be handed off to another client that wishes to use the tape.

Tape Queue Method

This field may be set to either `FIFO` or `Offset`. If set to `FIFO`, the Core Server will process tape requests per VV in a FIFO fashion without doing any intelligent ordering. This somewhat mimics legacy behavior and can be useful for sites which want to closely manage their own ordering. If set to `Offset`, the Core Server will attempt to reorganize the VV session queue in order to improve performance. If available, this may use drive technologies such as Recommended

Access Ordering (RAO).

If a file called `nodrive` exists in the HPSS tmp folder (HPSS_PATH_TMP), its presence will disable RAO-based sorting of tape requests and instead use strict offset ordering. If a file called `noschedule` exists in this folder it will disable all scheduling, resulting in a FIFO ordering for tape requests from that point until the file is removed. Note that this will not impact any existing schedules.

Individual VVs may also be toggled to FIFO scheduling using the **FIFO** check box on the VV information window. See the Core Server Tape Volume information window for more information.

Tape Queue Active Set Size

Number of sessions which may be scheduled together. This yields a compromise between scheduling efficiency and potential wait times since this size is the maximum number of items within that set that may be scheduled ahead of any other request in that set. This may also be limited by the hardware limitations of Recommended Access Ordering (RAO) if the drive a tape is mounted on supports RAO.

Tape Max Reader Wait (seconds)

This is the amount of time, in seconds, that readers of a tape may spend waiting on writers before the scheduling priority shifts to prefer readers over writers for some period of time. There is a minimum of five minutes between reader/writer priority changes. Setting this to zero disables preference switching and will cause writers to always be preferred.

PVL Max Connection Wait (seconds)

The amount of time, in seconds, the Core Server will wait to connect to the PVL before declaring an error in pending PVL jobs.

Fragment Smallest Block

Fragment Smallest Block sets the boundary used to determine where to remove excess clusters from allocations that have excess space. The smallest block returned to the free space map from the excess space at the end of an extent will be this size or larger.

Log ENOSPACE Details (debug only)

If ON, the Core Server logs detailed debug class log messages about disk and tape allocation requests that end in HPSS_ENOSPACE. In normal operation, these details are too voluminous to log economically, so they are not logged by default. But if the cause of an ENOSPACE condition is not evident, setting this switch will enable logging of all of the details. Because of the volume of information logged, leaving this switch on for long periods of time is not recommended.

Root User ID

The numerical ID for the root user used by the Core Server.

Root is Superuser

If this flag is ON, clients using the root user ID will have superuser powers and permissions.

I/O Operations to Log (debug only)

Classes of IODs/IORs to Log (debug only)

These switches control optional logging in the Core Server. The Core Server always logs IODs and IORs when I/O operations fail, but does not log IODs and IORs for successful operations. These switches allow the administrator to turn on logging of IODs and IORs in successful tape and disk I/O operations for diagnostic purposes.

Normally these switches should be off to avoid logging large amounts of information that won't be useful. Changes to these switch settings in the managed object take effect immediately. Reinitialization of the server is not necessary.

Subsystem Statistics tab

Stages

The number of stage requests processed in the Core Server since startup or last reset of the statistics.

Migrations

The number of migrate requests processed in the Core Server since startup or last reset of the statistics.

Purges

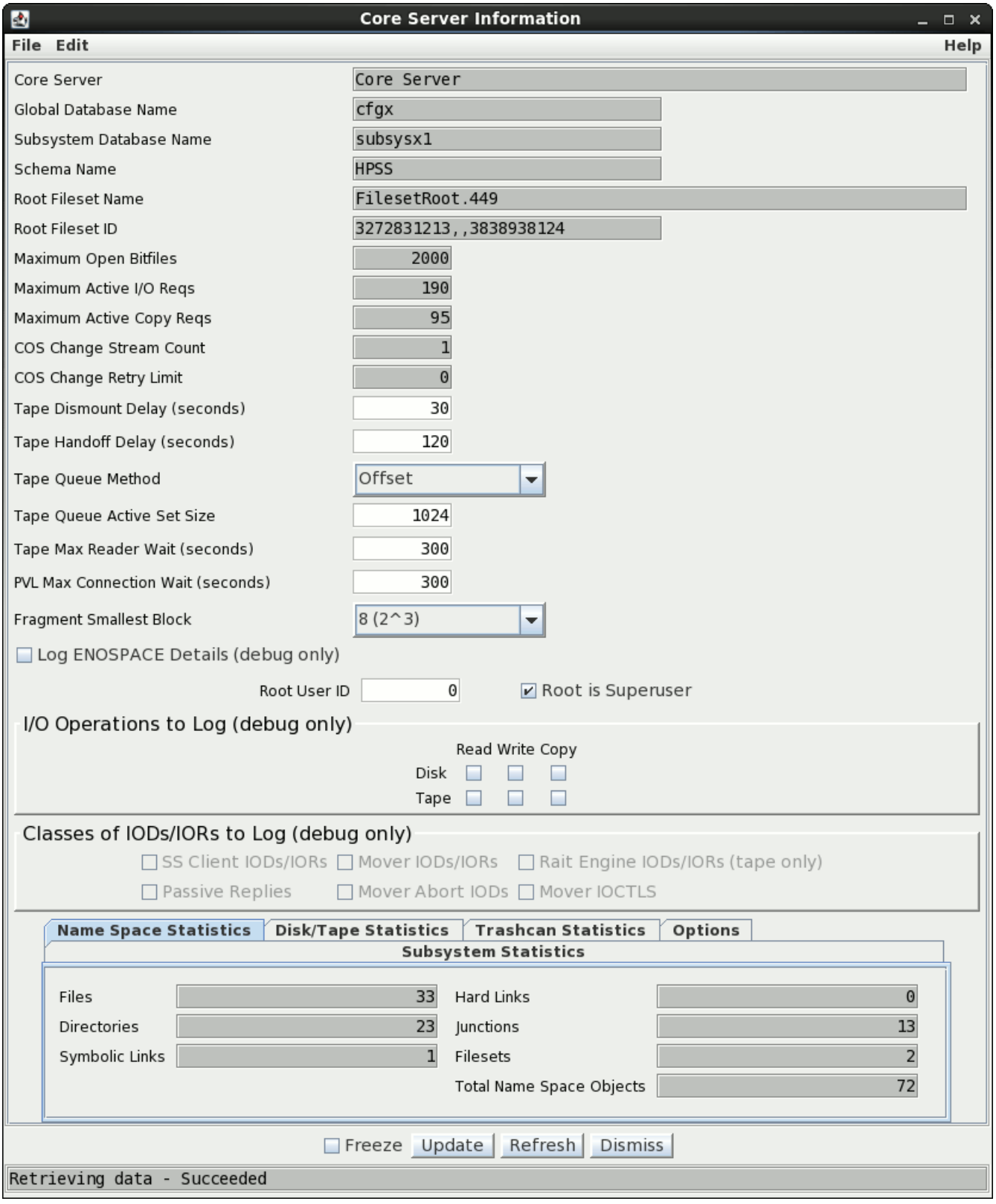
The number of purge requests processed in the Core Server since startup or last reset of the statistics.

File Deletes

The number of bitfile delete requests processed in the Core Server since startup or last reset of the statistics.

Last Reset Time

The last time the subsystem statistics were reset. If this value is "0", the statistics have not been reset since server startup.



Name Space Statistics tab

Files

The number of files managed by this Core Server.

Directories

The number of directories managed by this Core Server.

Symbolic Links

The number of symbolic links managed by this Core Server.

Hard Links

The number of hard links managed by this Core Server.

Junctions

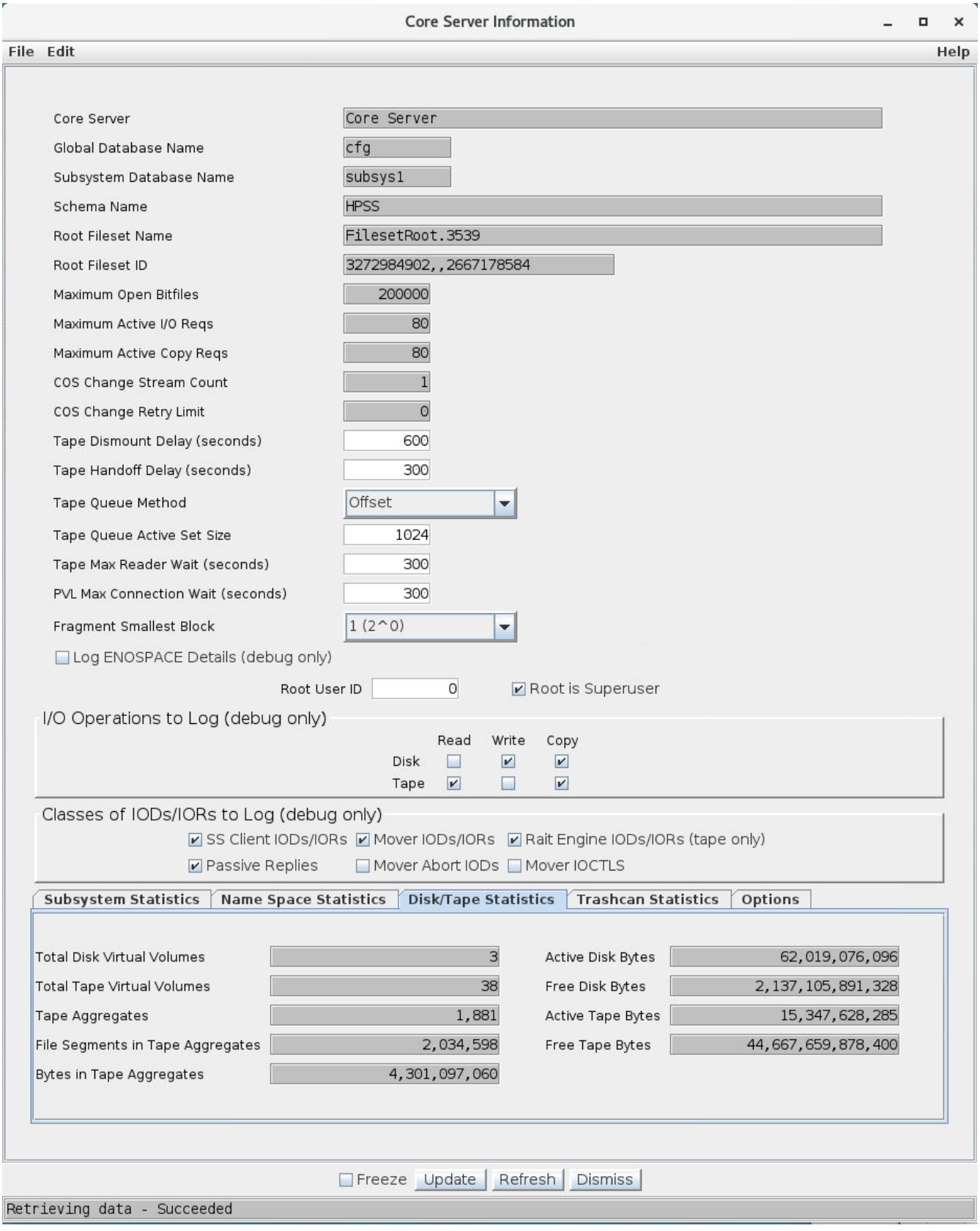
The number of junctions managed by this Core Server.

Filesets

The number of filesets managed by this Core Server.

Total Name Space Objects

The total number of name space records in the database. Note that this total may not match the total achieved by summing the number of the above objects. Some objects, such as hard links, require more than one record.



Disk/Tape Statistics tab

Total Disk Virtual Volumes

The number of disk virtual volumes managed by this Core Server.

Total Tape Virtual Volumes

The number of tape virtual volumes managed by this Core Server.

Tape Aggregates

The number of tape aggregates managed by this Core Server. A tape aggregate is a single tape storage segment that contains more than one file.

File Segments in Tape Aggregates

The number of file segments in tape aggregates managed by this Core Server. This is a count of files within all aggregates managed by the server. Files may be split across tape aggregates so may be counted multiple times in this statistic.

Bytes in Tape Aggregates

The number of bytes stored in the tape aggregates managed by this Core Server.

Active Disk Bytes

This is the sum of the bytes in the storage segments assigned to files, including slack space. Slack space is the disk space assigned to disk storage segments that is not occupied with file data. The sum of the lengths of files on disk will be less than the **Active Disk Bytes** because of the presence of slack space.

Free Disk Bytes

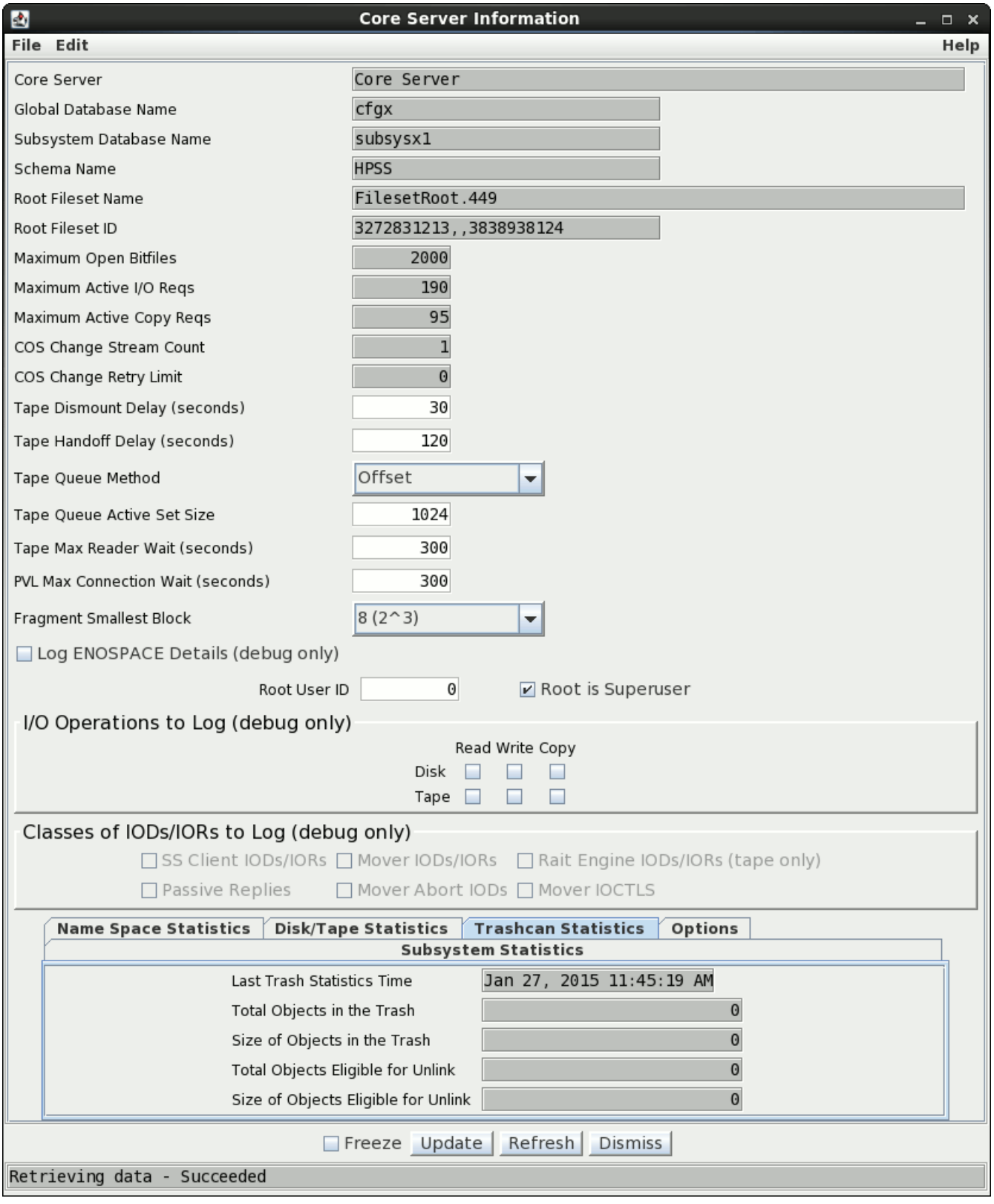
This is the sum of the bytes not assigned to files. More precisely, it is the sum of the disk volume bytes not assigned to disk storage segments.

Active Tape Bytes

This is the sum of the bytes assigned to files on tapes. It does not include deleted file space that has not been reclaimed.

Free Tape Bytes

This is an estimate based on the sum of the estimated sizes of the partially written and unwritten tape volumes. It is not, and cannot be, an accurate value as the amount of data that can be written on tapes varies with individual tape volumes and data compression levels.



Trashcan Statistics tab

Last Trash Statistics Time

The last time that the background thread ran which periodically gathers trashcan-related statistics.

Total Objects in the Trash

This is a count of the total number of entries currently in all trashcans (the NSTRASH database table).

Size of Objects in the Trash

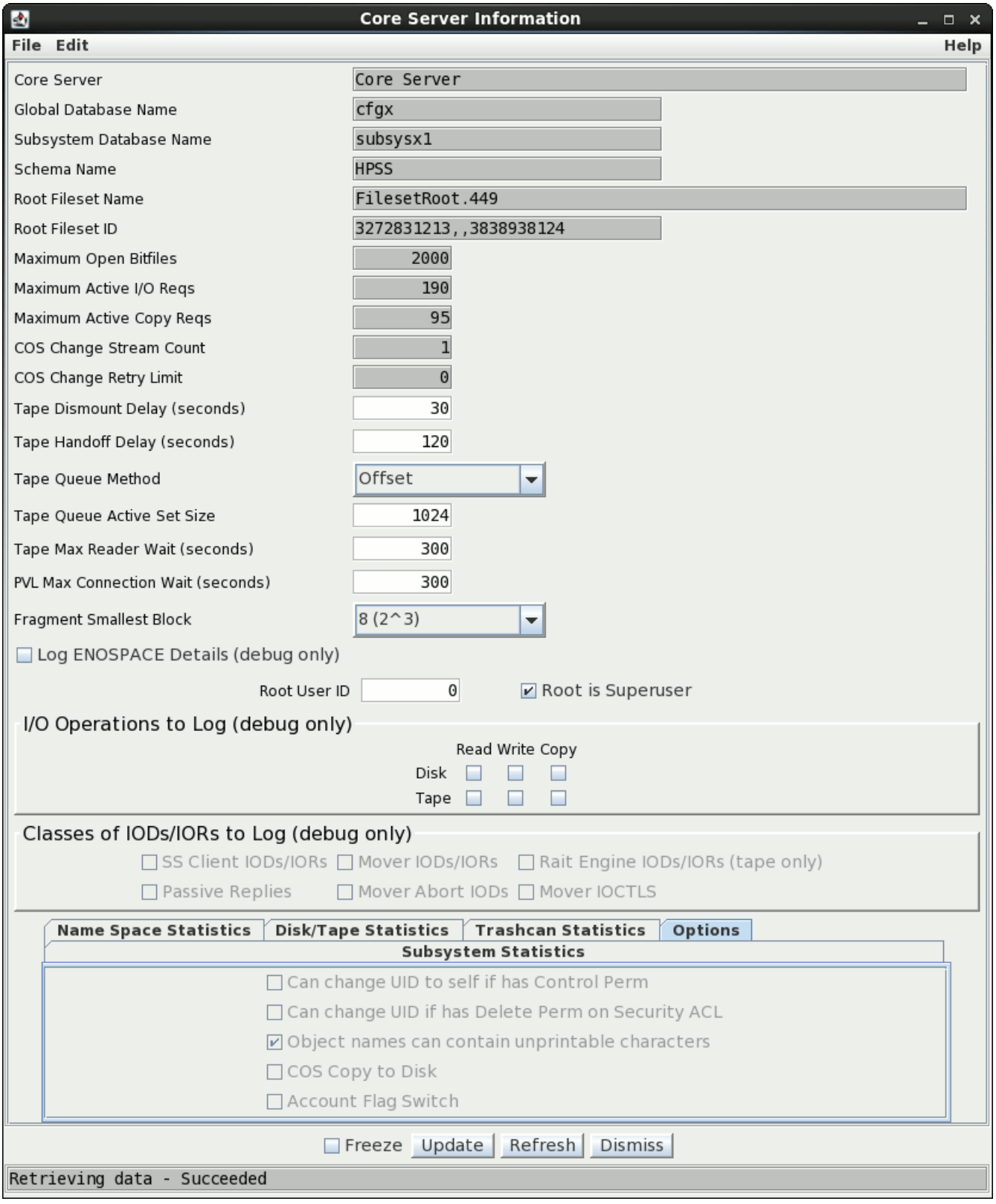
The total number of bytes used by all files currently in trashcans (the NSTRASH database table).

Total Objects Eligible for Unlink

The count of the number of entries in the NSTRASH table that are currently eligible for permanent deletion from trashcans (the NSTRASH table). Entries in trashcans become eligible for deletion when they have been in a trashcan for **Trash Unlink Eligible Time** seconds. **Trash Unlink Eligible Time** is a field on the *Global Configuration* window. (Refer to Global Configuration window.)

Size of Objects Eligible for Unlink

The total number of bytes used by the trashcan file objects that are currently eligible for permanent deletion.



Options tab

Can change UID to self if has Control Perm

If this flag is ON, any user having Control permission to an object can change the UID of that object to their own (but not any other) UID. This field can be changed only from the *Global Configuration* window. It is displayed here for informational purposes only.

Can change UID if has Delete Perm on Security ACL

If this flag is ON, any principal found on the Core Server Security ACL having Delete (and only

Delete) permission may change the UID of an object they own to any other UID. This field can be changed only from the *Global Configuration* window. It is displayed here for informational purposes only.

Object names can contain unprintable characters

If this flag is ON, no check will be made to ensure that new object and fileset names contain only printable characters. If this flag is OFF, new object and fileset names must be composed of only printable characters and if any unprintable characters are detected in these new names, an error will be returned. Printable characters are defined as all characters in the range from 0x20 to 0x7E (inclusive) in the 7-bit ASCII character set. This field can be changed only from the *Global Configuration* window. It is displayed here for informational purposes only.

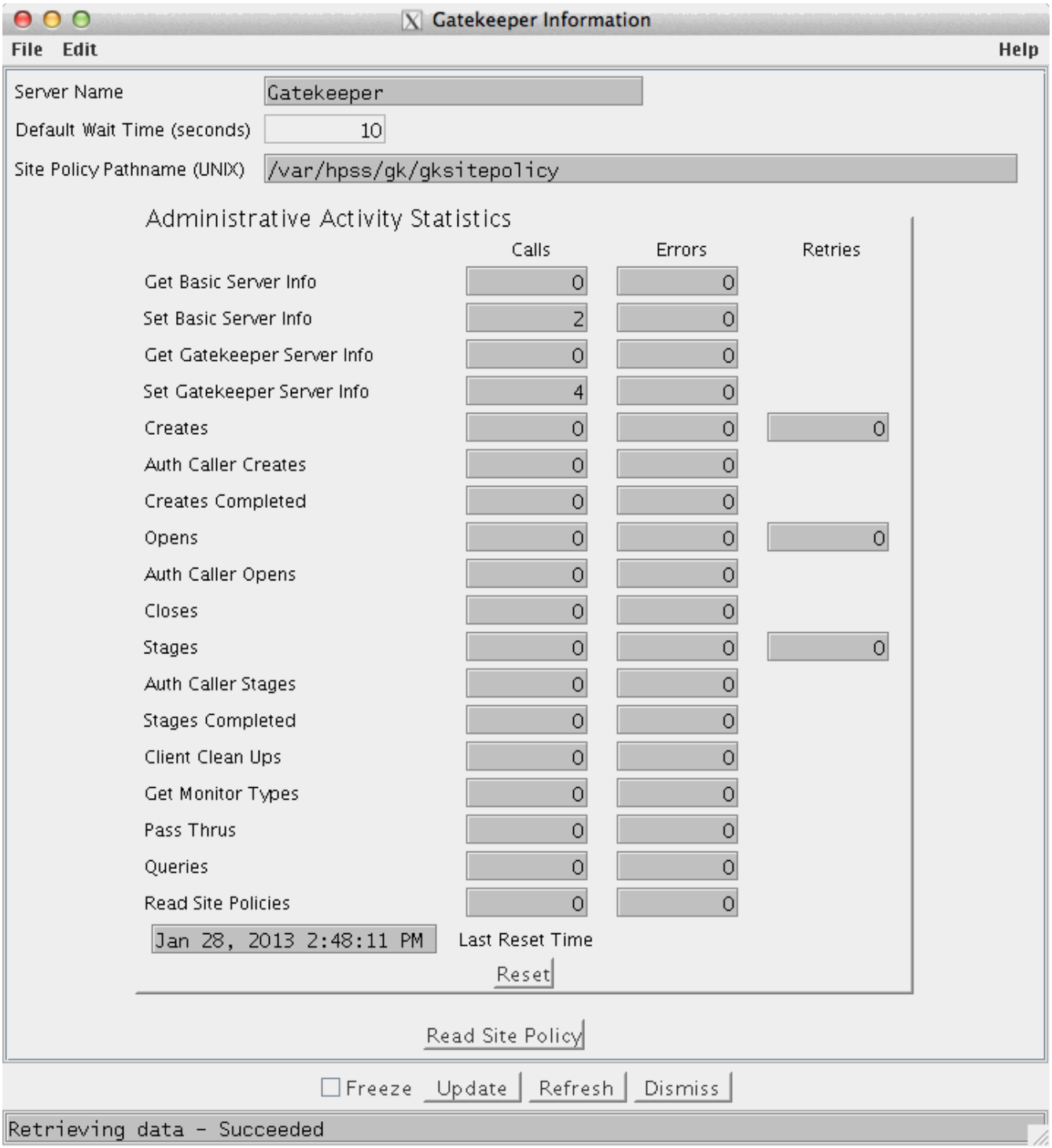
COS Copy to Disk

If ON, all copy operations associated with COS changes should be directed to disk if the hierarchy of the target COS has a level 0 disk storage class. This flag may be changed only from the Specific tab of the *Core Server Configuration* window. It is displayed here for informational purposes only.

Account Flag Switch

If this flag is ON, it indicates to use the second accounting bit in the bitfile metadata table to indicate that the bitfile has valid accounting metadata. If the flag is OFF, it indicates to use the first bit. The default is OFF. This flag setting is changeable only via special procedures. It is provided to allow a site to recover or rebuild its accounting metadata if this becomes necessary. Special procedures are required for this. Contact HPSS support to do this.

16.3.2.2. Gatekeeper Information window



Field descriptions

Server Name

The descriptive name of the Gatekeeper.

Default Wait Time (seconds)

The number of seconds the client must wait before retrying a request. This value must be greater than zero and is used if the Gatekeeping Site Interface returns a wait time of zero for the create, open, or stage request being retried. Changing the value of this field will cause the Gatekeeper to use the new value until the next restart at which point it will then go back to using the value defined in the *Gatekeeper Configuration* window. Refer to Gatekeeper-specific configuration.

Site Policy Pathname (UNIX)

This field can only be set from the *Gatekeeper Configuration* window. Refer to Gatekeeper-specific configuration.

Administrative activity statistics

These fields are internal volatile statistics for the Gatekeeper. Each row describes a different API. There are three columns: Calls, Errors and Retries. Each of these columns represent a count of the number of calls, errors, and retries since the statistics were reset. The Calls column is the number of times the API was called. The Errors column is the number of times the API call resulted in an error being returned. The Retries column is the number of times that the API call resulted in the HPSS_ERETRY error being returned to the Client API. The Retries column only applies to API requests that may return the HPSS_ERETRY status. A retry is not counted as an error.

The statistics are reset to zero whenever the Gatekeeper is recycled or the **Reset** button is clicked.

Get Basic Server Info

Statistics from the gk_admin_ServerGetAttrs API.

Set Basic Server Info

Statistics from the gk_admin_ServerSetAttrs API. This API is called by the SSM System Manager when the Gatekeeper's *Basic Server Information* window is opened.

Get Gatekeeper Server Info

Statistics from the gk_admin_GKGetAttrs API.

Set Gatekeeper Server Info

Statistics from the gk_admin_GKSetAttrs API. This API is called by the SSM System Manager when the *Gatekeeper Information* window is opened.

Creates

Statistics from the gk_Create API. This API is called by the Core Server when the Gatekeeper is monitoring Create Requests.

Auth Caller Creates

Statistics from authorized caller (for example, MPS) calls to the gk_Create API.

Creates Completed

Statistics from the gk_CreateComplete API. This API is called by the Core Server when the Gatekeeper is monitoring Create Requests and the create completes.

Opens

Statistics from the gk_Open API. This API is called by the Core Server when the Gatekeeper is monitoring Open Requests.

Auth Caller Opens

Statistics from authorized caller (for example, MPS) calls to the gk_Open API.

Closes

Statistics from the gk_Close API. This API is called by the Core Server when the Gatekeeper is monitoring Open Requests and the file is closed.

Stages

Statistics from the gk_Stage API. This API is called by the Core Server when the Gatekeeper is monitoring Stage Requests.

Auth Caller Stages

Statistics from authorized caller (for example, MPS) calls to the gk_Stage API.

Stages Completed

Statistics from the gk_StageComplete API. This API is called by the Core Server when the Gatekeeper is monitoring Stage Requests and the stage completes.

Client Clean Ups

Statistics from the gk_CleanUpClient API. This API is called by the Core Server when the Gatekeeper is monitoring Requests and a client disconnects.

Get Monitor Types

Statistics from the gk_GetMonitorTypes API. This API is called by the Core Server to figure out what types of Requests being monitored by the Gatekeeper.

Pass Thrus

Statistics from the gk_PassThru API.

Queries

Statistics from the gk_Query API.

Read Site Policies

Statistics from the gk_ReadSitePolicy API.

Last Reset Time

The time stamp when the statistics were last reset to zero.

Associated button descriptions

Reset

Reset the administrative activity statistics to "0" and the **Last Reset Time** to the current date and time.

Read Site Policy

This button is used to inform the GatekeepingSite Interface (gk_site_ReadSitePolicy) of a policy change to the **Site Policy Pathname** file.

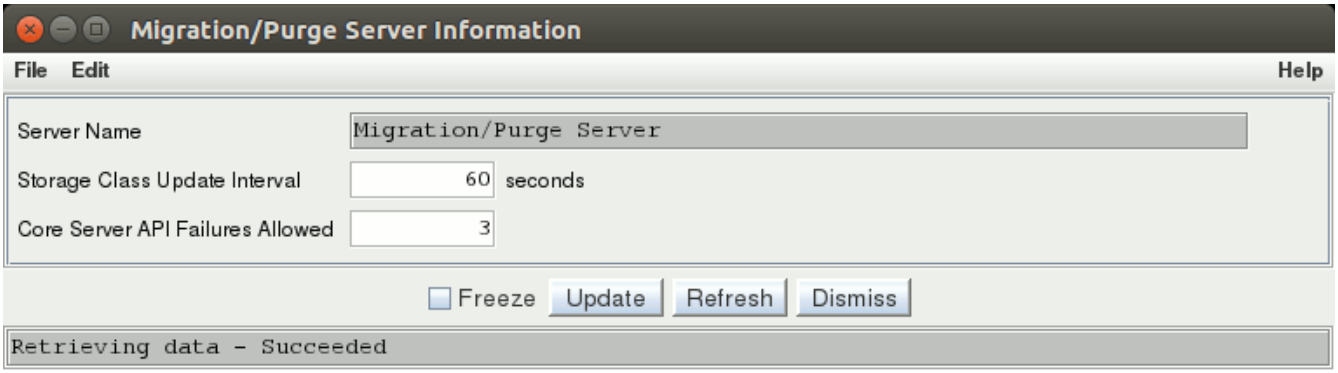
Note: a "policy change" cannot include a change in the types of requests being monitored. When changing the types of requests being monitored (authorized caller, create, open, and stage), the Gatekeeper must be recycled so that the Core Server can learn which request types need to

include Gatekeeping services.

16.3.2.3. Migration/Purge Server Information window

This window allows you to view and update the type-specific information associated with a Migration/Purge Server (MPS).

Any changes made to fields on this window are sent directly to the MPS and are effective immediately.



Field descriptions

Server Name

The descriptive name of the MPS.

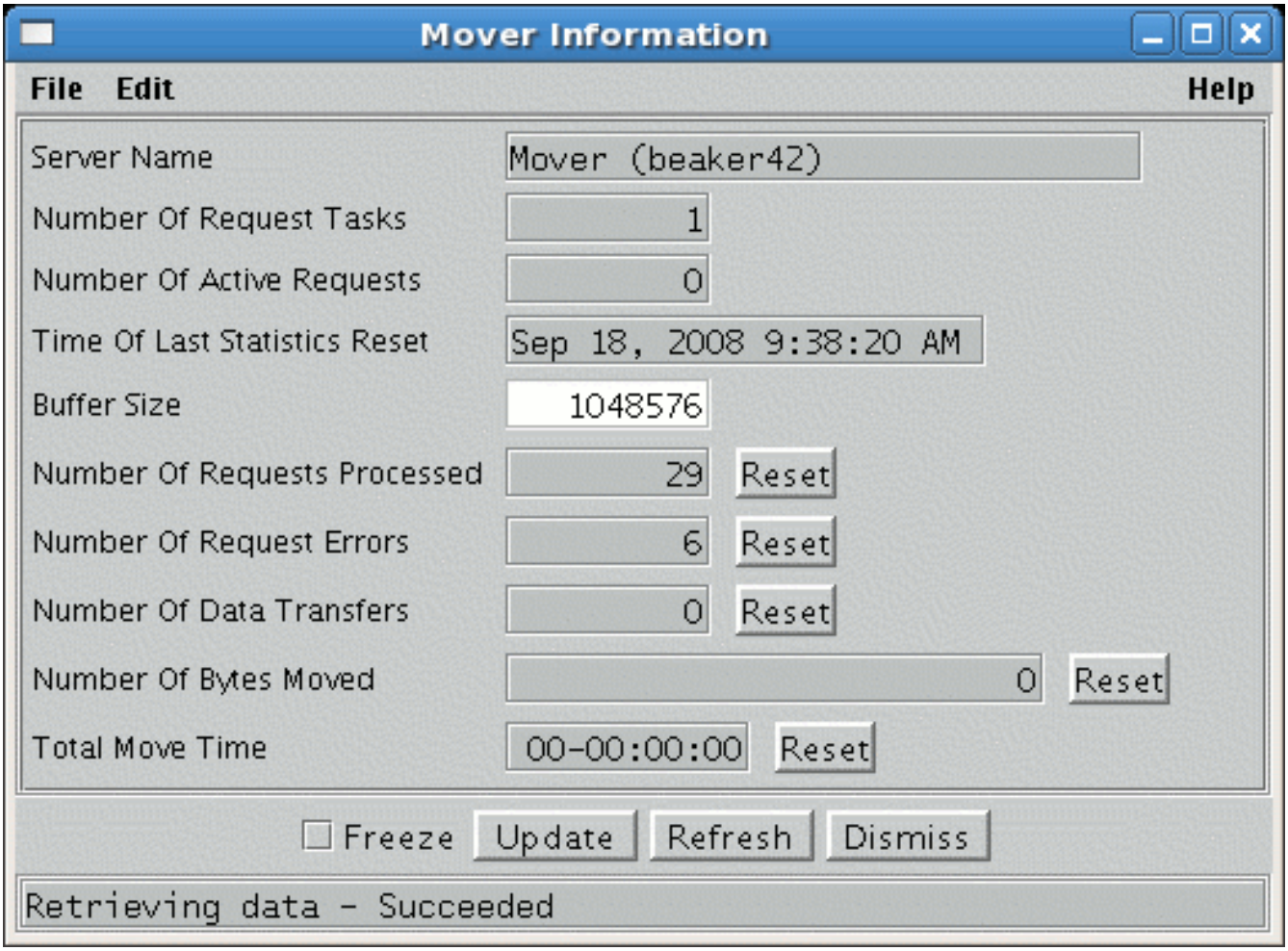
Storage Class Update Interval

The interval that indicates how often the MPS will query the Core Server in its subsystem to get the latest storage class statistics. This is also the interval the MPS uses to check whether it needs to initiate a purge operation on the storage class based on the associated purge policy. The valid range for this field is 10 - 600 seconds.

Core Server API Failures Allowed

The maximum number of consecutive Core Server API failures allowed before MPS aborts a migration or purge run. This applies to both disk and tape migration as well as disk purge. If this limit is reached during a disk migration run, migration skips to the next hierarchy to which that disk storage class belongs. Once the run has attempted to migrate all such hierarchies the run will end. If this limit is reached during a disk purge or tape migration run, the migration or purge run aborts.

16.3.2.4. Mover Information window



This window allows you to view and update the type-specific information associated with a Mover. Any changes made to fields on this window are sent directly to the Mover after the appropriate button is clicked and are effective immediately.

Field descriptions

Server Name

The descriptive name of the Mover.

Number of Request Tasks

The number of Mover request-processing tasks that currently exist.

Number of Active Requests

The number of active requests that are being handled by the Mover.

Time of Last Statistics Reset

The time and date when the Mover statistics were last reset.

Buffer Size

The size, in bytes, of each buffer used by the Mover when transferring data. Each Mover task uses two buffers of this size to perform double buffering during I/O operations. See the *Mover Configuration* window **Buffer Size** field described in Mover-specific configuration for more detailed information.

Number of Requests Processed

The number of requests that the Mover has processed since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Number of Request Errors

The number of requests that have returned errors since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Number of Data Transfers

The number of data transfers that the Mover has handled since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Number of Bytes Moved

The number of bytes of data that the Mover has transferred since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Total Move Time

The total time, in days, hours, minutes, and seconds, that the Mover tasks have spent transferring data since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

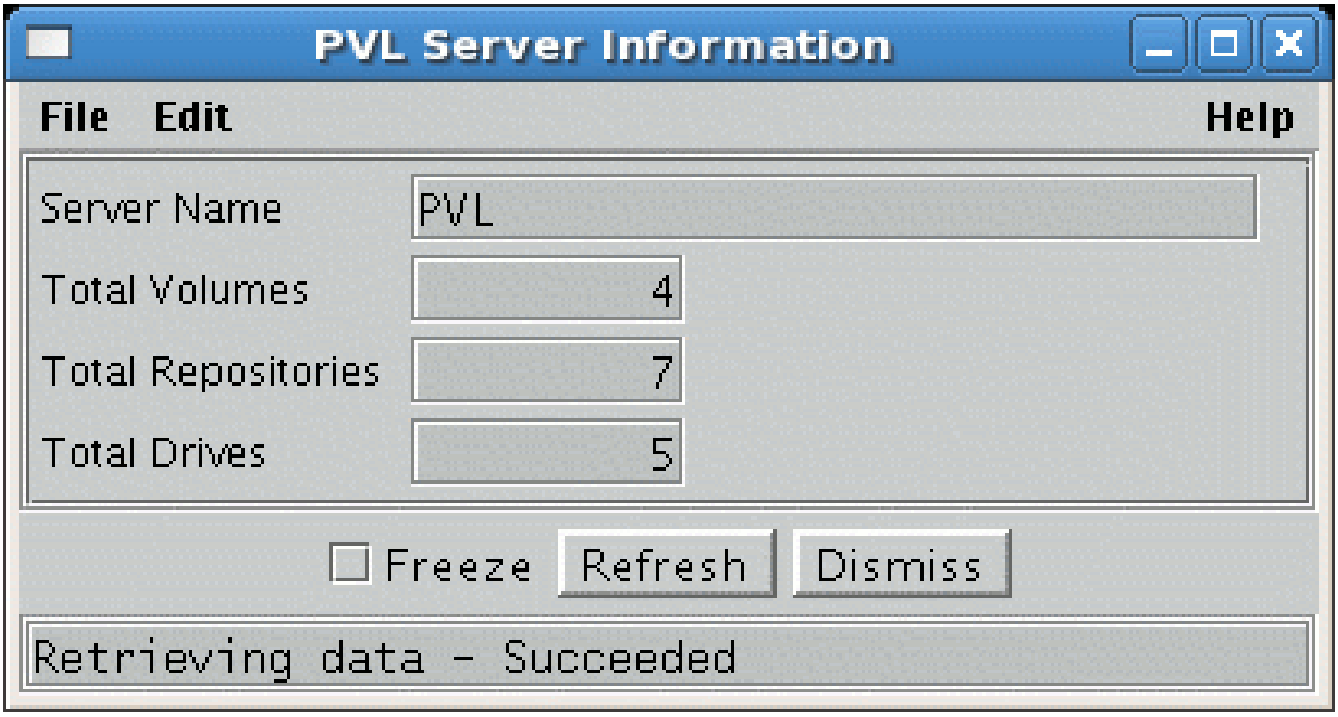
Operational notes:

1. The Mover statistics (**Number of Requests Processed**, **Number of Request Errors**, **Number of Data Transfers**, **Number of Bytes Moved**, and **Total Move Time**) may all be reset to zero. If any of these fields are reset the **Time of Last Statistics Reset** field will also be updated with the current time. The statistics fields are all initialized to zero when the Mover is started.
2. The **Total Move Time** statistic should not be used to determine Mover throughput rates if the Mover handles striped media, since this is a total for all Mover tasks involved in the transfer. This total will most likely be considerably greater than the elapsed time necessary to transfer the data.

Related information

HPSS Error Manual : *Problem diagnosis and resolution* chapter : *Mover problems* section.

16.3.2.5. Physical Volume Library (PVL) information window



This window allows you to view the type-specific information associated with a PVL.

Field descriptions

Server Name

The descriptive name of the PVL.

Total Volumes

The total number of volumes that have been imported into the PVL.

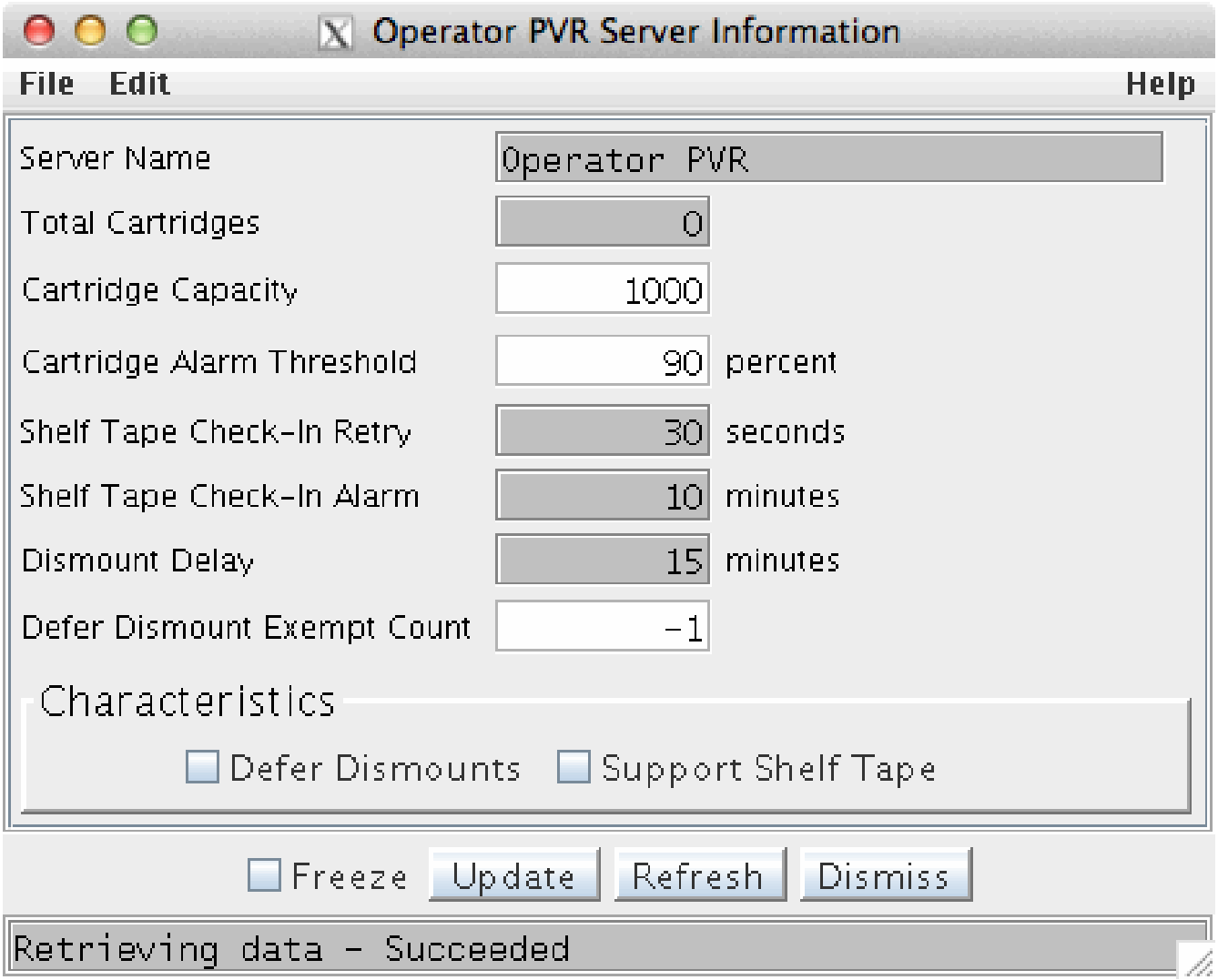
Total Repositories

The total number of PVRs in the *Servers* list window.

Total Drives

The total number of drives controlled by this PVL.

16.3.2.6. Operator PVR information window



This window allows you to view and update the type-specific information associated with an Operator PVR. Any changes made to fields on this window are sent directly to the PVR and are effective immediately.

Field descriptions

Server Name

The descriptive name of the PVR.

Total Cartridges

The number of cartridges currently being managed by the PVR.

Cartridge Capacity

The total number of cartridge slots in the library dedicated to this HPSS PVR. This may or may not be the total cartridge capacity of the library; a site might use part of the library for some other HPSS PVR or for some non-HPSS application. The PVR uses the **Cartridge Capacity** field and the **Cartridge Alarm Threshold** field to determine when to send an alarm that the total cartridge threshold has been exceeded.

Cartridge Alarm Threshold

The percentage of the **Cartridge Capacity** at which the PVR will send an alarm.

Shelf Tape Check-In Retry

The number of seconds the PVR will wait before asking the robot if a requested shelf tape has been checked in. The PVR will continue checking at this interval until the tape is checked in. This field applies only if the **Support Shelf Tape** check box is selected. The retry value must be "30" or greater.

Shelf Tape Check-In Alarm

The PVR will periodically log alarm messages when a requested shelf tape has not been checked in. This field specifies the number of minutes between alarms. This field is only active if the **Support Shelf Tape** check box is selected. The alarm value must be "2" or greater.

Dismount Delay

When **Defer Dismounts** is checked, this value is used by the PVL to determine the number of minutes that dismounts are delayed after the last data access.

Defer Dismount Exempt Count

If set greater than "0", this value will specify the number of drives within a PVR that will be exempt from the defer dismount logic. This behavior will take place for all drives within the PVR with a drive/device "Library Unit ID" set (see Drive/Device Configuration). If multiple Library Unit IDs are set, the same Exempt count will apply to all Library Unit ID subsets.

The goal of this option is to allow a finer control of the defer dismount logic to account for transfer costs (for example, elevator, pass through costs). Defer dismount logic to apply logic on a sub library case (unit). The definition of a "unit" is placed in the hands of the administrator and likely based on library characteristics and topology.

Characteristics

Flags for the PVR.

Defer Dismounts

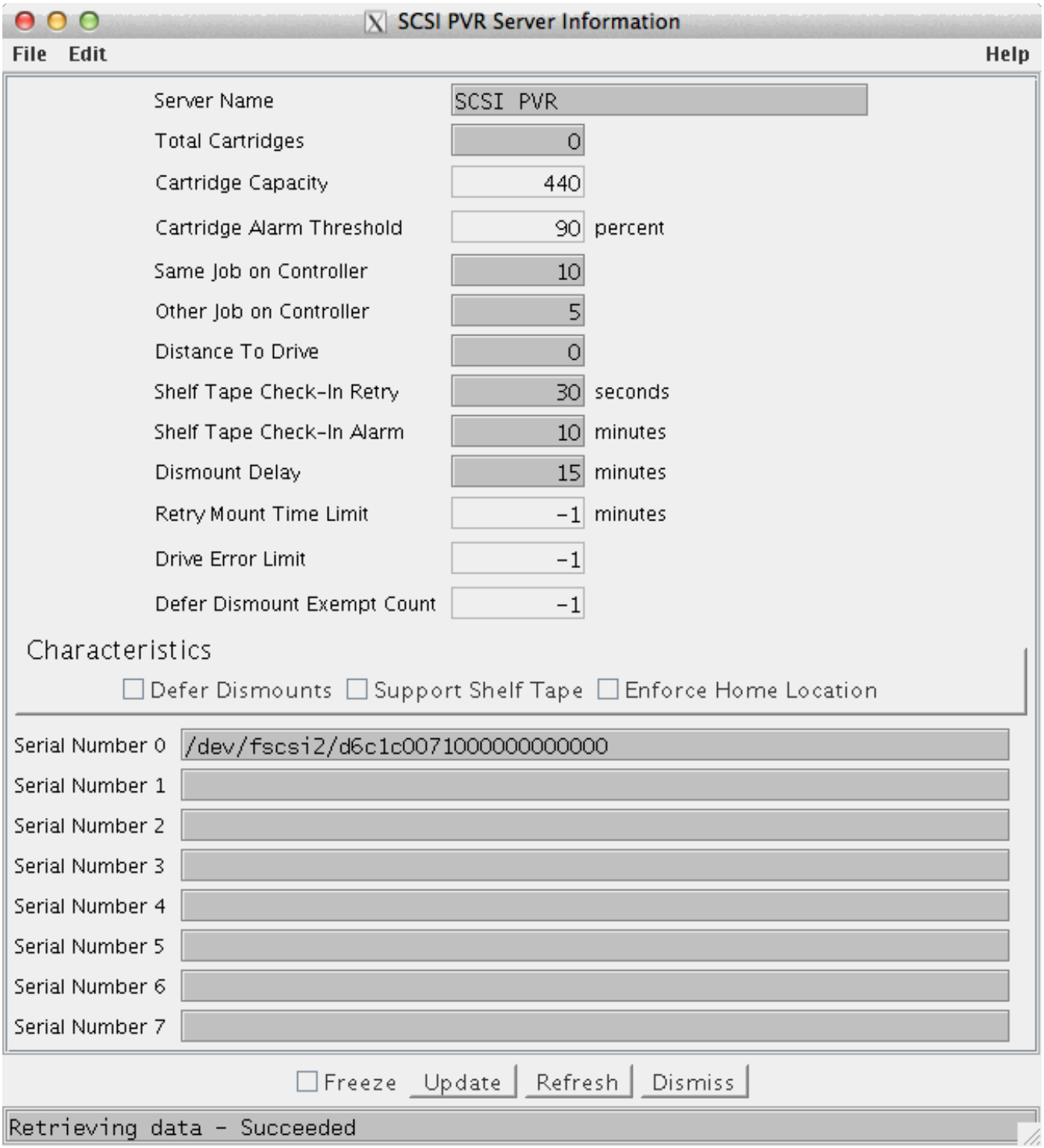
If ON, the PVL will delay the dismounting of a tape cartridge until the drive is required by another job or until the **Dismount Delay** time limit is exceeded. If the **Defer Dismount Exempt Count** is also set, the defer dismount logic will not be exercised on the number of drives set. This may result in free drives available for new mount requests without first generating defer dismount operations. See **Defer Dismount Exempt Count** for details.

A PVL restart is required in order for changes to this flag to take effect.

Support Shelf Tape

If ON, the PVR and the PVL will support the removal of cartridges from the tape library using the shelf_tape(7) utility.

16.3.2.7. SCSI PVR information window



This window allows you to view and update the type-specific information associated with a SCSI PVR. Any changes made to fields on this window are sent directly to the PVR and are effective immediately.

Field descriptions

Server Name

The descriptive name of the PVR.

Total Cartridges

The number of cartridges currently being managed by the PVR.

Cartridge Capacity

The total number of cartridge slots in the library dedicated to this HPSS PVR. This may or may not be the total cartridge capacity of the library; a site might use part of the library for some other HPSS PVR or for some non-HPSS application. The PVR uses the **Cartridge Capacity** field and the **Cartridge Alarm Threshold** field to determine when to send an alarm that the total cartridge threshold has been exceeded.

Cartridge Alarm Threshold

The percentage of the **Cartridge Capacity** at which the PVR will send an alarm.

Same Job on Controller, Other Job on Controller, Distance To Drive

These three values are used by the PVR when selecting a drive for a tape mount operation. The three values are essentially weights that are used to compute an overall score for each possible drive. After the score has been calculated, the drive with the lowest score is selected for the mount. If two or more drives tie for the lowest score, one drive is selected at random. The score is calculated as follows:

> Score =
> Weight 1 × Cartridges from this job mounted on this drive's controller +
> Weight 2 × Cartridges from other jobs mounted on this drive's controller +
> Weight 3 × Units of distance from the cartridge to the drive

This method has the effect of distributing a striped tape mount across as many controllers as possible for the best performance. It also will try to pick controllers that are currently driving a minimum number of tapes. So, in an environment with many tape drives per controller, the above algorithm will minimize the load on any one controller.

The **Distance To Drive** helps minimize mount times by mounting the tape in a physically close drive. All other things being equal, the tape will be mounted in the closest drive. However, if the closest drive is attached to a heavily used controller, then a more distant drive will be selected.

Shelf Tape Check-In Retry

The number of seconds the PVR will wait before asking the robot if a requested shelf tape has been checked in. The PVR will continue checking at this interval until the tape is checked in. This field applies only if the **Support Shelf Tape** check box is selected. The retry value must be "30" or greater.

Shelf Tape Check-In Alarm

The PVR will periodically log alarm messages when a requested shelf tape has not been checked in. This field specifies the number of minutes between alarms. This field is only active if the **Support Shelf Tape** check box is selected. The alarm value must be "2" or greater.

Dismount Delay

When **Defer Dismounts** is checked, this value is used by the PVL to determine the number of minutes that dismounts are delayed after the last data access.

Retry Mount Time Limit

The default value for this field is "-1". When the default value is used, if an error is encountered during a PVR mount operation, the mount will pend and be retried every five minutes. Setting a value in this field will change the mount behavior to periodically retry the mount until the specified time limit is exceeded. Once exceeded, an error is generated and the mount request is canceled. If the mount request would have resulted in a write operation, the error returned will cause the Core Server to set the **VV Condition** of the associated tape volume to DOWN. Once in DOWN state, the volume will no longer be available for read or write operations. For further information about the Core Server VV Condition, see Core Server Tape Volume information window.

Drive Error Limit

This field is used in conjunction with the PVR Server **Retry Mount Time Limit**. If the number of consecutive mount errors which occur to any drive in this PVR equal or exceed this value, the drive is automatically locked by the PVL. The only mount errors that apply are those set through the **Retry Mount Time Limit** mechanism. The **Drive Error Count** field in the PVL Drive Information records the number of consecutive errors on a drive by drive basis. To turn off the automatic drive disable feature, set the **Drive Error Limit** to "0" or "-1". Changing its value will not change drive disable behavior until the PVL is recycled.

Defer Dismount Exempt Count

If set greater than "0", this value will specify the number of drives within a PVR that will be exempt from the defer dismount logic. This behavior will take place for all drives within the PVR with a drive/device "Library Unit ID" set (see Drive/Device Configuration). If multiple Library Unit IDs are set, the same Exempt count will apply to all Library Unit ID subsets.

The goal of this option is to allow a finer control of the defer dismount logic to account for transfer costs (for example, elevator, pass through costs). Defer dismount logic to apply logic on a sub library case (unit). The definition of a "unit" is placed in the hands of the administrator and likely based on library characteristics and topology.

Characteristics

Flags for the PVR.

Defer Dismounts

If ON, the PVL will delay the dismounting of a tape cartridge until the drive is required by another job or until the **Dismount Delay** time limit is exceeded. If the **Defer Dismount Exempt Count** is also set, the defer dismount logic will not be exercised on the number of drives set. This may result in free drives available for new mount requests without first generating defer dismount operations. See **Defer Dismount Exempt Count** for details.

A PVL restart is required in order for changes to this flag to take effect.

Support Shelf Tape

If ON, the PVR and the PVL will support the removal of cartridges from the tape library using the shelf_tape(7) utility.

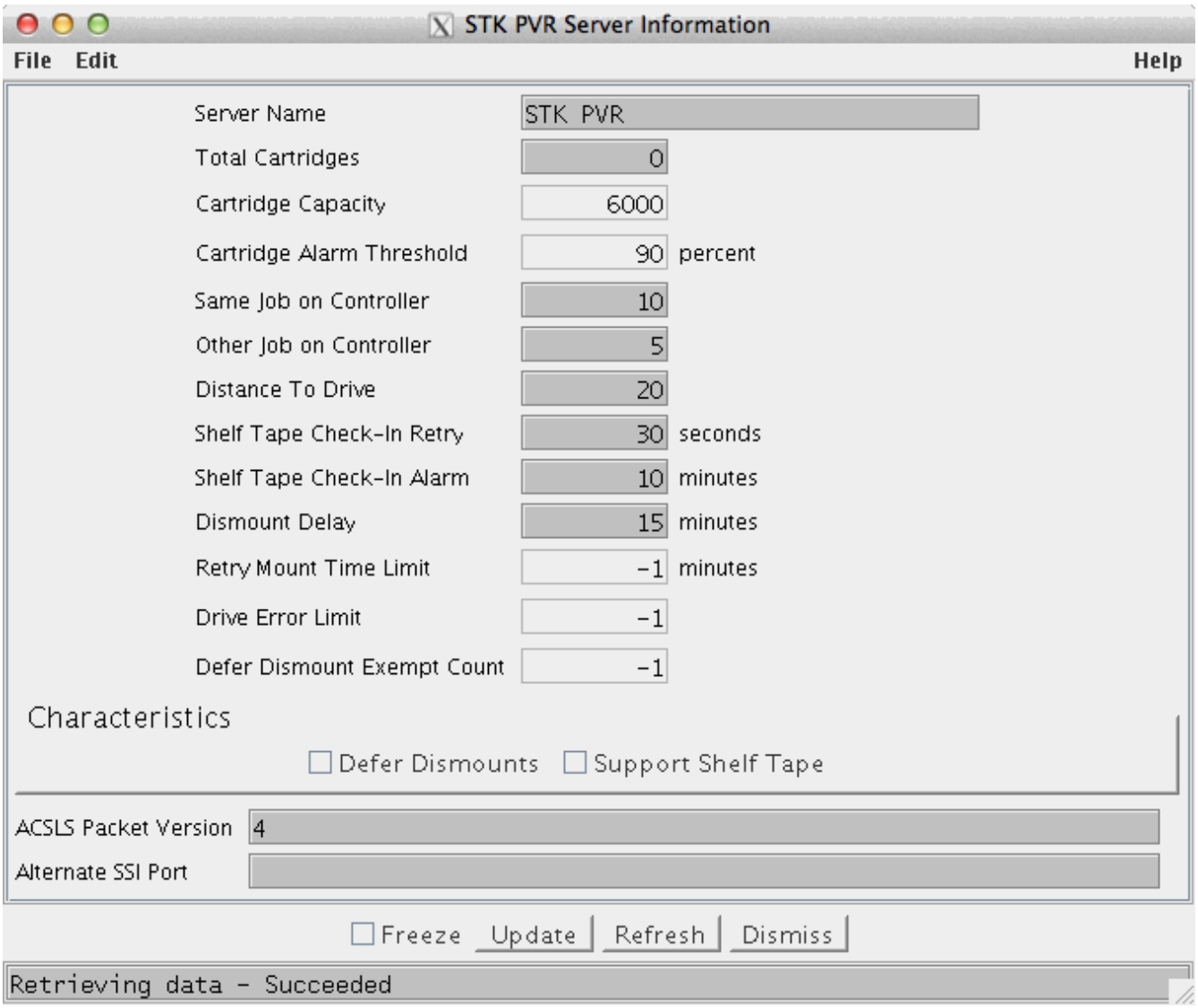
Enforce Home Location

If ON, the SCSI PVR will always try to dismount a mounted cart back to its home location. Otherwise, it will just use the first free slot. The scsi_home(7) utility can be used to view the home location values.

Serial Number 0 through Serial Number 7

The serial numbers of the robots the SCSI PVR is controlling. Used to detect control paths to the robots. Only Serial Number 0 is valid at this time.

16.3.2.8. STK PVR information window



This window allows you to view and/update the type-specific information associated with a STK PVR. Any changes made to fields on this window are sent directly to the PVR and are effective immediately.

Field descriptions

Server Name

The descriptive name of the PVR.

Total Cartridges

The number of cartridges currently being managed by the PVR.

Cartridge Capacity

The total number of cartridge slots in the library dedicated to this HPSS PVR. This may or may not be the total cartridge capacity of the library; a site might use part of the library for some other HPSS PVR or for some non-HPSS application. The PVR uses the **Cartridge Capacity** field and the **Cartridge Alarm Threshold** field to determine when to send an alarm that the total cartridge threshold has been exceeded.

Cartridge Alarm Threshold

The percentage of the **Cartridge Capacity** at which the PVR will send an alarm.

Same Job on Controller, Other Job on Controller, Distance To Drive

These three values are used by the PVR when selecting a drive for a tape mount operation. The three values are essentially weights that are used to compute an overall score for each possible drive. After the score has been calculated, the drive with the lowest score is selected for the mount. If two or more drives tie for the lowest score, one drive is selected at random. The score is calculated as follows:

> Score =
> Weight 1 × Cartridges from this job mounted on this drive's controller +
> Weight 2 × Cartridges from other jobs mounted on this drive's controller +
> Weight 3 × Units of distance from the cartridge to the drive

This method has the effect of distributing a striped tape mount across as many controllers as possible for the best performance. It also will try to pick controllers that are currently driving a minimum number of tapes. So, in an environment with many tape drives per controller, the above algorithm will minimize the load on any one controller.

The **Distance To Drive** helps minimize mount times by mounting the tape in a physically close drive. For STK drives, a unit of distance is from one Silo/LSM (Library Storage Module) to the next. This means that the tape must go through a pass-through port or elevator. This process is more time consuming. Therefore the default value for STK robots is a higher value which forces the tape to stay in the same Silo/LSM even if it means the drive selected is attached to a controller which is heavily used. All other things being equal, the tape will be mounted in the closest drive.

Shelf Tape Check-In Retry

The number of seconds the PVR will wait before asking the robot if a requested shelf tape has been checked in. The PVR will continue checking at this interval until the tape is checked in. This field applies only if the **Support Shelf Tape** check box is selected. The retry value must be "30" or greater.

Shelf Tape Check-In Alarm

The PVR will periodically log alarm messages when a requested shelf tape has not been checked in. This field specifies the number of minutes between alarms. This field is only active if the

Support Shelf Tape check box is selected. The alarm value must be "2" or greater.

Dismount Delay

When **Defer Dismounts** is checked, this value is used by the PVL to determine the number of minutes that dismounts are delayed after the last data access.

Retry Mount Time Limit

The default value for this field is "-1". When the default value is used, if an error is encountered during a PVR mount operation, the mount will pend and be retried every five minutes. Setting a value in this field will change the mount behavior to periodically retry the mount until the specified time limit is exceeded. Once exceeded, an error is generated and the mount request is canceled. If the mount request would have resulted in a write operation, the error returned will cause the Core Server to set the **VV Condition** of the associated tape volume to DOWN. Once in DOWN state, the volume will no longer be available for read or write operations. For further information about the Core Server VV Condition, see Core Server Tape Volume information window.

Drive Error Limit

This field is used in conjunction with the PVR Server **Retry Mount Time Limit**. If the number of consecutive mount errors which occur to any drive in this PVR equal or exceed this value, the drive is automatically locked by the PVL. The only mount errors that apply are those set through the **Retry Mount Time Limit** mechanism. The **Drive Error Count** field in the PVL Drive Information records the number of consecutive errors on a drive by drive basis. To turn off the automatic drive disable feature, set the **Drive Error Limit** to "0" or "-1". Changing its value will not change drive disable behavior until the PVL is recycled.

Defer Dismount Exempt Count

If set greater than "0", this value will specify the number of drives within a PVR that will be exempt from the defer dismount logic. This behavior will take place for all drives within the PVR with a drive/device "Library Unit ID" set (see Drive/Device Configuration). If multiple Library Unit IDs are set, the same Exempt count will apply to all Library Unit ID subsets.

The goal of this option is to allow a finer control of the defer dismount logic to account for transfer costs (for example, elevator, pass through costs). Defer dismount logic to apply logic on a sub library case (unit). The definition of a "unit" is placed in the hands of the administrator and likely based on library characteristics and topology.

Characteristics

Flags for the PVR.

Defer Dismounts

If ON, the PVL will delay the dismounting of a tape cartridge until the drive is required by another job or until the **Dismount Delay** time limit is exceeded. If the **Defer Dismount Exempt Count** is also set, the defer dismount logic will not be exercised on the number of drives set. This may result in free drives available for new mount requests without first generating defer dismount operations. See **Defer Dismount Exempt Count** for details.

A PVL restart is required in order for changes to this flag to take effect.

Support Shelf Tape

If ON, the PVR and the PVL will support the removal of cartridges from the tape library using the shelf_tape(7) utility.

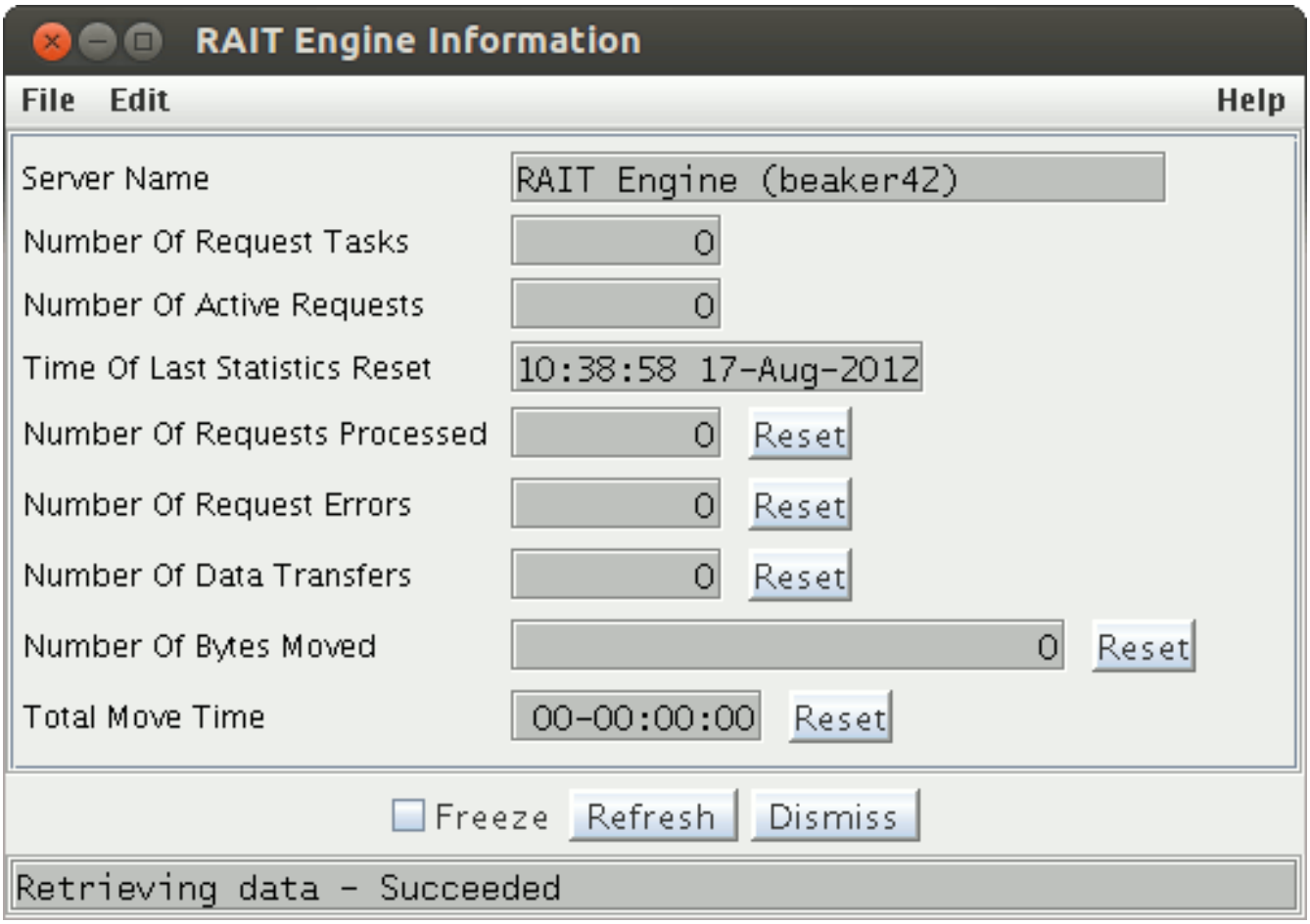
ACSLS Packet Version

The packet version used by STK's ACSLS software.

Alternate SSI Port

A site running two SSI (Server System Interface) services on the same platform will specify the non-default port number of the second service via this field. Refer to STK PVR additional information.

16.3.2.9. RAIT Engine Information window



This window allows you to view and update the type-specific information associated with a RAIT Engine. Any changes made to fields on this window are sent directly to the RAIT Engine after the appropriate button is clicked and are effective immediately.

Field descriptions

Server Name

The descriptive name of the RAIT Engine.

Number of Request Tasks

The number of RAIT Engine request-processing tasks that currently exist.

Number of Active Requests

The number of active requests that are being handled by the RAIT Engine.

Time of Last Statistics Reset

The time and date when the RAIT Engine statistics were last reset.

Number of Requests Processed

The number of requests that the RAIT Engine has processed since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Number of Request Errors

The number of requests that have returned errors since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Number of Data Transfers

The number of data transfers that the RAIT Engine has handled since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Number of Bytes Moved

The number of bytes of data that the RAIT Engine has transferred since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero. *(Note: this field is not implemented in HPSS Release 7.4.1.)*

Total Move Time

The total time, in days, hours, minutes, and seconds, that the RAIT Engine tasks have spent transferring data since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the value to zero.

Operational notes

1. The RAIT Engine statistics (**Number of Requests Processed**, **Number of Request Errors**, **Number of Data Transfers**, **Number of Bytes Moved**, and **Total Move Time**) may all be reset to zero. If any of these fields are reset the **Time of Last Statistics Reset** field will also be updated with the current time. The statistics fields are all initialized to zero when the RAIT Engine is started.
2. The **Total Move Time** statistic should not be used to determine RAIT Engine throughput rates if more than one request is handled by the RAIT Engine concurrently. Since this is a total for all RAIT Engine tasks involved in the transfer. This total will most likely be considerably greater than the elapsed time necessary to transfer the data.

Related information

HPSS Error Manual : *Problem diagnosis and resolution* chapter : *Mover problems* section.

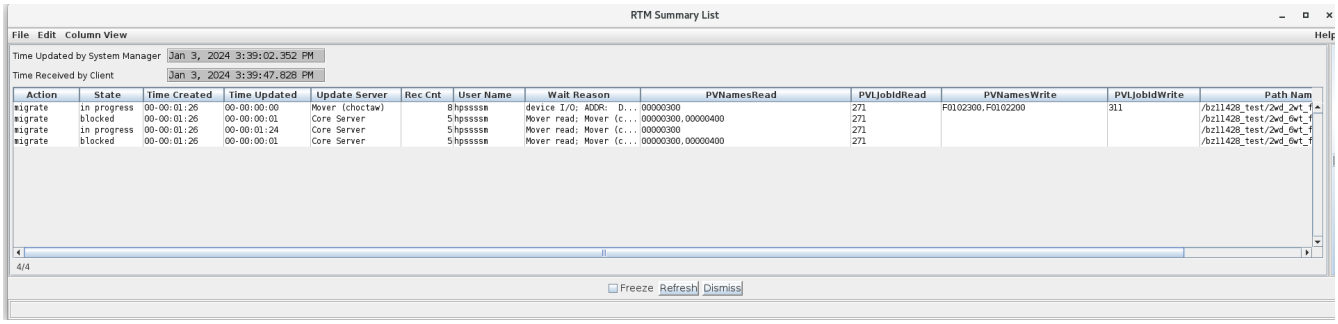
16.4. Real-Time Monitoring (RTM)

Real-time Monitoring (RTM) is a facility built into the Gatekeeper, the Core Server, the RAIT Engine, and the Mover. Whenever an HPSS thread acting on behalf of a user job or a high-level system

activity needs to block or process an operation that may take some time to complete, it will create an RTM record with information about what it is currently doing or waiting for. The RTM utility (rtmu(7)) can be used to query the list of the current requests and obtain pertinent data for each request. The RTM requests can be used to construct an overview of the current operations being processed by the HPSS system as well as to monitor the overall health and status of the system. RTM requests that remain in the system for an extended period of time may indicate that the system is under heavy load. If a request remains for an unusually long time, it may indicate a hang condition or a deadlock situation that may require special intervention.

HPSS administrators and operators may use SSM to view the active RTM requests. The *RTM Summary* window lists a summary of the current RTM requests. The *RTM Detail* window displays detailed information for selected RTM requests. Requests in the *RTM Summary* window can be aborted using the **Abort I/O** button. Refer to I/O Abort.

16.4.1. RTM Summary List



Field descriptions

RTM Summary List

This is the main portion of the window which displays various information about each RTM request summary.

ID

The RTM request identifier.

Action

The action or operation that this request is currently executing. Examples include "Mover write", "PVL verify", or "write tm" (that is, write a tape mark).

State

The state of the requested operation. Examples include `in progress`, `suspended`, or `blocked`.

SubsystemId

The ID of the HPSS subsystem owning this request.

Time Created

Of all the RTM request records with this ID, this field holds the age of the oldest. The time delta is the age, or time difference between when the request entered the system and the *RTM Summary List* window was last updated.

Time Updated

Of all the RTM requests records with this ID, this field holds the age of the request most recently updated by its server. The time delta is the age, or time difference between when the request was updated by the server and the *RTM Summary List* window was last updated.

Create Server

The server holding the RTM request with the oldest age or Time Created.

Update Server

The server holding the RTM request with the most recent update age or Time Updated.

Rec Cnt

The number of RTM records with this ID.

User Name

The printable username associated with this request (if available).

Path Name

The HPSS pathname of the file associated with this request (if available). The reporting level of the path name is controlled by the environment setting HPSS_RTM_USE_FULL_PATH. Refer to the entry for HPSS_RTM_USE_FULL_PATH in the HPSS Environment Variables for details.

Wait Reason

A text string indicating what the request is currently waiting for. It may also contain additional information on the resources being waited for.

PVLJobIdRead

The PVL job ID (if available) for the PV names being read from.

PVNamesRead

The names of the Physical Volumes associated with this read request (if available). If more than five PV names are involved, only the first five will be shown and a ",*" will be shown afterward.

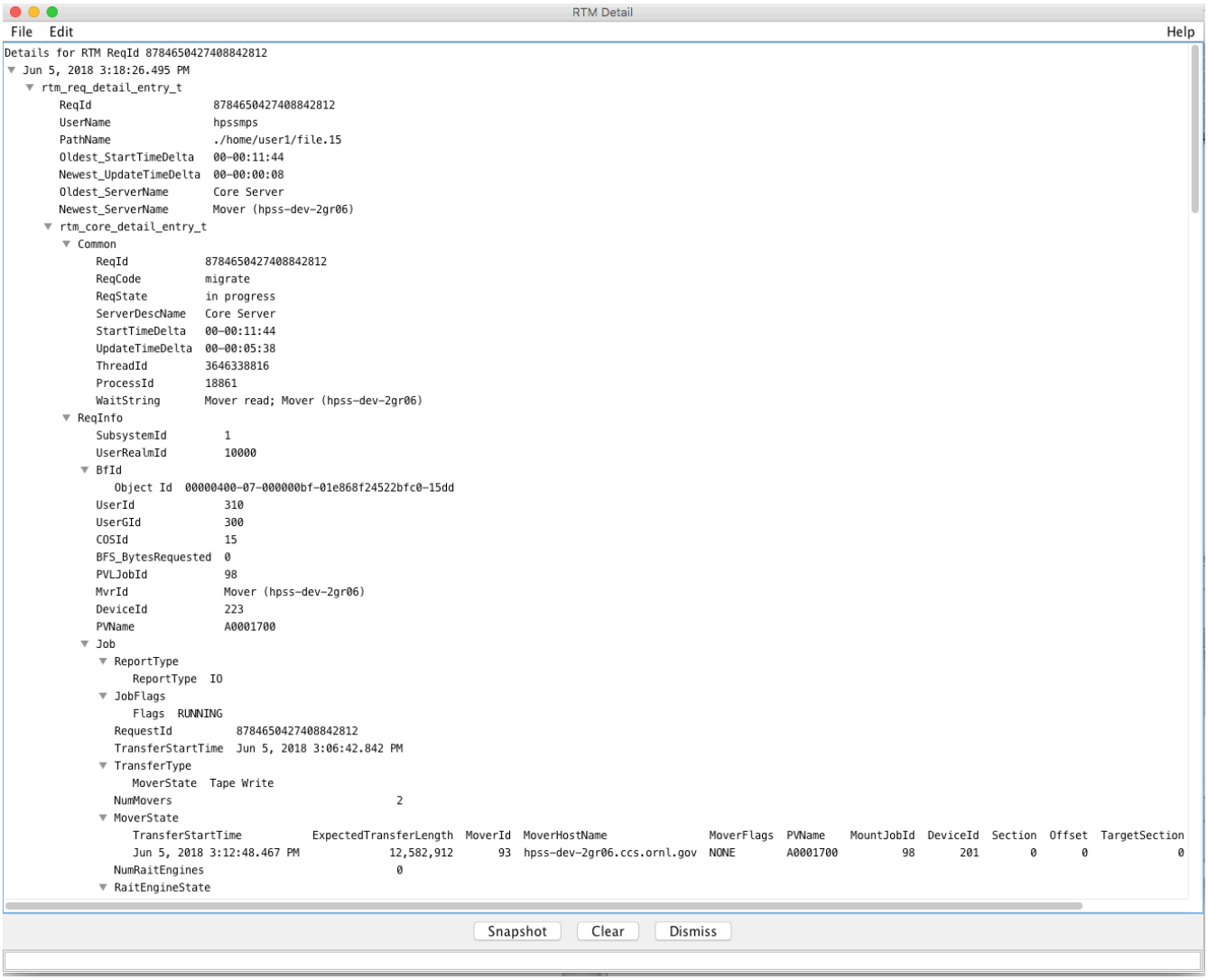
PVLJobIdWrite

The PVL job ID (if available) for the PV names being written to.

PVNamesWrite

The names of the Physical Volumes associated with this write request (if available). If more than five PV names are involved, only the first five will be shown and a ",*" will be shown afterward.

16.4.2. RTM Detail



The *RTM Detail* window displays a snapshot of the details of the selected RTM requests from the *RTM Summary List* window. This may contain information from multiple servers, Gatekeeper, Core, RAIT Engine, and Mover. The actual data displayed will be different for each server type and is displayed in a tree structure. Each node of the tree can be expanded or collapsed by clicking the mouse on the tree node indicator. A new snapshot can be taken and added to the display by clicking the **Snapshot** button. The tree can be cleared by clicking the **Clear** button.

Each detailed snapshot taken for an RTM request will contain the date and time the snapshot was taken at the top of the tree. Below the date and time there will be an **rtm_req_detail_entry_t.**

The fields contained in the **rtm_req_detail_entry_t** structure are:

ReqId

The RTM request identifier.

UserName

The printable username associated with this request (if available).

PathName

The HPSS pathname of the file associated with this request (if available). The reporting level of the path name is controlled by the environment setting HPSS_RTM_USE_FULL_PATH. Refer to the entry for HPSS_RTM_USE_FULL_PATH in the HPSS Environment Variables for details.

Oldest_StartTimeDelta

Of all the RTM request records with this **ReqId**, this field holds the age of the oldest when the snapshot was taken.

Newest_UpdateTimeDelta

Of all the RTM requests records with this **ReqId**, this field holds the age of the request most recently updated by its server when the snapshot was taken.

Oldest_ServerName

The server holding the RTM request with the oldest age or **Oldest_StartTimeDelta**.

Newest_ServerName

The server holding the RTM request with the most recent update age or **Newest_UpdateTimeDelta**.

Each server type that generates RTM request records (Gatekeeper, Core, RAIT Engine, and Mover) has a server-specific RTM data structure which will be displayed as a tree node below the **rtm_req_detail_entry_t** node. These structures are only displayed if there is a server of that type involved in the RTM request. There will be multiple entries if there are more than one server involved in the RTM request.

The server-specific RTM records are called:

- **rtm_gk_detail_entry_t** (Gatekeeper)
- **rtm_core_detail_entry_t** (Core)
- **rtm_mvr_detail_entry_t** (Mover)
- **rtm_rait_detail_entry_t** (RAIT Engine)

The server-specific data structures each have a **Common** and **ReqInfo** structure embedded in them. The **Common** structure will contain the same fields for each server type.

The Common fields are:

ReqId

The RTM request identifier.

ReqCode

The action or operation that this request is currently executing. Examples include `Mover write`, `PVL verify`, or `write tm` (that is, write a tape mark).

ReqState

The state of the requested operation. Examples include `in progress`, `suspended`, or `blocked`.

ServerDescName

The descriptive name of the server that holds this RTM request record.

StartTimeDelta

The age of this request since it entered the server. That is, the time difference between when

the request entered and when this snapshot was taken.

UpdateTimeDelta

The time difference between when this server last updated this RTM record and when this snapshot was taken.

ThreadId

The ID of the thread processing this request, useful in attaching to the thread to examine its state in the case of a hung request.

ProcessId

The ID of the UNIX process that is processing this request, useful in attaching to the process to examine its state in the case of a hung request.

WaitString

A text string indicating what the request is currently waiting for inside this server.

The **ReqInfo** portions of the server-specific data structures will contain different fields depending on the server type. Not all fields for a particular server-specific **ReqInfo** structure will be displayed. Only those fields that have valid data in them for the particular state of the RTM request will be displayed. The possible fields for each server-specific **ReqInfo** are listed below.

The Core Server ReqInfo structure may contain any or none of the following fields:

SubsystemId

The ID of the HPSS subsystem owning this request.

UserRealmId

The realm ID of the user associated with this request.

FilesetId

The ID of the fileset of the file associated with this request.

FilesetName

The text string name of the fileset of the file associated with this request.

BfId

The bitfile ID of the file associated with this request.

BfPathname

The HPSS pathname of the file associated with this request.

UserId

The ID of the user associated with this request.

UserGId

The group ID of the user associated with this request.

COSId

The Class Of Service ID of the file associated with this request.

BFS_SClassId

The storage class ID known to the bitfile server of the file segment associated with this request.

BFS_BytesRequested

The number of bytes to be moved in this request as known by the bitfile server.

BFS_BytesMoved

The number of bytes already moved for this request as known by the bitfile server.

PVLJobId

The ID of the PVL job associated with this request.

MvrId

The ID of the Mover this request is currently waiting on.

DeviceId

The device this request is currently waiting on to complete a data move operation.

Segment

The ID of the storage segment being operated on by this request.

VV

The ID of the virtual volume associated with this request.

PVName

The name of the physical volume associated with this request.

CurrentRelPosition

The current media position of this physical volume as a relative address.

CurrentAbsPosition

The current media position of this physical volume as an absolute address.

ExcessVVsInUse

The tape virtual volumes in use by this storage class have exceeded its quota. Waiting for a this number of virtual volumes to become available.

FamilyId

The ID of the file family of the file associated with this request.

SS_SClassId

The storage class ID known to the storage server of the file/segment associated with this request.

SS_BytesRequested

The number of bytes to be moved by this request as known by the storage server.

SS_BytesMoved

The number of bytes that have been moved for this request as known by the storage server.

Job

A structure which describes the current job tracking information for this request. Its fields are described below:

ReportType

The type of report being delivered. The values are: `IO`, `VV Mount`, `VV Handoff`, `Tape Positioning`, `Tape Alignment`, and `Unknown`.

JobFlags

Flags which describe the status of the job. The values are `RUNNING`, `ABORTING`, and `ABORTED`. The job is in the `ABORTING` state when an abort action is being processed, and once that action has been completed it changes to be `ABORTED`.

RequestId

The request ID.

TransferStartTime

The time that the job began.

TransferType

The type of job being executed.

- `None` (No type)
- `Disk Check` (Check the disk size)
- `Disk Read` (Reading from a disk)
- `Disk Write` (Writing to a disk)
- `Disk Copy Read` (Reading from a disk for a copy)
- `Disk Copy Write` (Writing to a disk for a copy)
- `Disk Mount` (Mounting a disk volume)
- `Tape Read` (Reading from a tape)
- `Tape Write` (Writing to a tape)
- `Tape Verify` (Verify a tape volume)
- `Tape Mount` (Mounting a tape volume)
- `Tape Assignment` (Waiting to be assigned a tape)
- `Tape Position` (Positioning a tape volume)
- `Tape Alignment` (Aligning tape volume)
- `Tape Recommended Access Order` (Waiting on an RAO request to a drive)

- `Tape Session Queue` (Queued session against a volume)
- `BFS Scheduling` (Waiting on bitfile to be accessible)
- `BFS Background Stage Queue` (Waiting in the background stage queue)
- `BFS Running` (Running)

MoverState

Information about a request to a Mover participating in the job. There may be multiple **MoverState** structures included.

TransferStartTime: When the request to the Mover began.

ExpectedTransferLength: The length of the I/O, in bytes.

MoverId: The server ID of the Mover.

MoverHostName: The hostname of the Mover TCP process.

MoverFlags: Information about the state of the I/O request in this Mover.

PVName: The physical volume associated with the Mover.

MountJobId: The PVL job ID.

DeviceId: The device ID being used for the operation.

PVPosition: For positioning requests, the location to position to.

TargetSection: For positioning requests, the section to position to.

AlignmentError: Whether a tape alignment error occurred.

RAITEngineState

Information about a request to a RAIT Engine participating in the job. There may be multiple **RAITEngineState** structures included.

TransferStartTime: When the request to the RAIT Engine began.

ExpectedTransferLength: The length of the I/O, in bytes.

REId: The server ID of the RAIT Engine.

REHostName: The hostname of the RAIT Engine process.

REDescName: The descriptive name of the RAIT Engine.

REFlags: Information about the state of the I/O request in this RAIT Engine.

PVLState

Information about a request to the PVL in the job. There may be multiple **PVLState** structures included.

PVName: The physical volume involved in the operation.

PVFlags: The status of the mount operation (`NOT-MOUNTED`, `MOUNT`, `MOUNTED`, `DEALLOC`, `ACTIVE`).

PVLJobId: The PVL job ID.

BFSState

Information about the BFS state for the job.

BfId: The bitfile identifier involved in the job.

JobDescription: A string describing what the job is doing in the BFS.

PendingJobId

PVL job ID.

The Mover ReqInfo structure may contain any or none of the following fields:

InitialTransferOffset

The offset of the start of the Mover device for the transfer associated with this request.

CurrentTransferOffset

The current offset of the Mover device for the transfer associated with this request.

DeviceId

The ID of the device associated with this request.

VolumeId

The name of the volume associated with this request.

BytesRequested

The number of bytes the Mover has been asked to transfer for this request.

BytesMoved

The number of bytes the Mover has already transferred for this request.

The RAIT Engine ReqInfo structure may contain any or none of the following fields:

InitialTransferOffset

The offset of the start of the transfer associated with this request.

CurrentTransferOffset

The current offset of the transfer associated with this request.

BytesRequested

The number of bytes the RAIT Engine has been asked to transfer for this request.

BytesMoved

The number of bytes the RAIT Engine has already transferred for this request.

ParityBufferIndex

The index number of the buffer that the parity thread is currently computing.

The Gatekeeper ReqInfo structure may contain any or none of the following fields:

AuthorizedCaller

Indicates whether the caller is authorized or not.

BitFileId

The bitfile ID of the file associated with this request.

ConnectionId

The ID of the (Core Server) connection associated with this request.

ClientConnectedId

The ID of the end client connection associated with this request.

ControlNo

A unique control ID used by the Gatekeeper server.

RealmId

The realm ID of the user associated with this request.

GroupId

The group ID of the user associated with this request.

HostAddr

The address of the originating host.

RequestType

The type of this request (`Open`, `Create`, or `Stage`).

Oflag

The Open flags associated with this file open request.

StageFlags

Flags associated with this file stage operation.

StageLength

The number of bytes to stage.

StageOffset

The offset of the file where the stage is to begin.

StageStorageLevel

The storage class level that the file will be staged to.

UserId

The ID of the user associated with this request.

WaitTime

The length of time this request has been waiting.

16.5. I/O Abort

This section describes how to abort I/O operations, and what to expect from HPSS following an abort request.

16.5.1. Aborting an I/O

I/O Abort applies to a number of HPSS operations; however, there are many operations which may be beyond its scope. The following types of requests can be aborted:

- Application I/O requests to disk or tape via any client interface
- Stages, including background stages and queued background stages
- Open requests which require a stage
- Migration operations including migrations from MPS or Force Migration requests from Recover
- Repack operations
- Logical Block Protection (LBP) Verify operations
- Requests for tape access waiting in the HPSS tape scheduler

Here are some examples of requests which fall outside of this capability and which cannot be aborted via this feature:

- A change COS request; the I/O can be canceled but the change COS request remains
- A long running name space operation such as a directory listing or a UDA query
- A PVR cartridge move operation
- A request to get file attributes which has become blocked due to a lock on the file
- Disk purge operations
- Creating or deleting HPSS storage resources

Requests for I/O can be aborted in several ways:

- Via the SSM GUI *RTM Summary* window
- Via the ADM command line interface
- Via the SSM GUI *Active Storage Classes* window (for migrations)
- Via applications that support aborting I/O via the Client API

To abort a request from the GUI, simply select one or more requests from the *RTM Summary* window, select the "Abort I/O" button, and confirm.

To abort a request from the ADM, use the **rtm cancel** command:

```
hpssadm> rtm cancel -id <request id>
```

Request IDs can be identified using the RTM summary or detail information with the HPSS ADM, GUI, or rtmu.

An abort request will return successfully when the HPSS Core Server has received the abort request and taken some action to cancel the I/O request. The abort request will return prior to the request being completed and leaving the RTM window.

In some cases, a request may be in a state where it is not abortable and will return with an error. This is likely due to the request being very early in its life and not registered as an abortable job, or not being a request which can be aborted because it is not related to an I/O request (for example, a Core Server tape handoff operation).

Issuing a **Suspend** or **Stop** on an active storage class migration will abort ongoing I/O. This will allow HPSS administrators to quickly suspend/stop migrations for maintenance operations.

16.5.2. Expectations in Aborting I/O

The primary use case for I/O Abort is in dealing with stuck or hung jobs; jobs which may never complete. Another significant use case is ending accidental work that has been spun up against the system; for example, an accidental large stage request, which may occupy the system for a significant amount of time for no purpose. For these use cases, I/O Abort tends to work very well. Because small I/O requests can very quickly enter and exit the system, it can be difficult to cancel ongoing small file I/O; however, this is not a situation that I/O Abort is intended to address. There are other methods for dealing with this.

Another important use case for I/O Abort is quickly quiescing migration activity in preparation for a downtime, or other maintenance work, by issuing a **Suspend** or **Stop** on each active storage class migration.

HPSS implements its abort interface as a "lazy" interface. The request to abort returns once the Core Server has received it and determined whether the operation associated with the incoming request ID can be aborted. This lazy method is employed because, depending upon the current state of the request, it could take several minutes for the request to ultimately wind down. For example, if a request for a stage is aborted and the tape is in the middle of being mounted the tape mount may still have to complete and then be dismounted before the request can end, which can take a significant amount of time.



It is important to consider that aborting a request may generate significant work, such as dismounting a large number of tape volumes which were in use by the aborted request. This could temporarily delay other work attempting to use the same library for other purposes.

Since the abort interface is "lazy", the request will continue to appear in RTM. However, in the case that some action has been taken to abort the request, the RTM details will list the job as being in an "ABORTING" or "ABORTED" state, depending on whether the abort action has already been taken or not.



Requests will remain in the *RTM Summary* window and continue to be updated until they are successfully aborted or they complete.

Aborted requests are logged differently in many cases. The abort request is logged as an EVENT message for every request that is aborted. In addition, resulting log messages that would normally be ALARM messages will generally be reduced to DEBUG messages.

From an HPSS client application perspective, an HPSS I/O request that has been aborted will appear as an EIO (Input/Output Error). Unlike some errors, the Client API will not retry an aborted request automatically. Client applications can retrieve more detailed information about the result from via the Client API and thereby distinguish a failed request from an aborted one.

16.6. Starting HPSS

This section describes how to start and stop HPSS servers and prerequisite software.

16.6.1. Starting HPSS prerequisite software

Invoke the `/opt/hpss/bin/rc.hpss` script as root to start the HPSS prerequisite software such as DB2, Kerberos, HPSS Login Service, and LDAP. By default, stdout and stderr from commands started by rc.hpss(7) are redirected to the logfile `HPSS_PATH_LOG/hpss_stdout_stderr.log` unless the **-o** option is used. Specifying the **-o** option to rc.hpss(7) will cause stdout and stderr of commands invoked within the script to be sent to `/dev/console`, the terminal window where rc.hpss(7) is running.

To start the prerequisite software, use the **-p** option to rc.hpss(7):

```
% su -
% /opt/hpss/bin/rc.hpss -p [start]
```

16.6.2. Starting HPSS servers

Most HPSS servers are started via SSM which calls an appropriate Startup Daemon to begin execution of the selected HPSS server. This means that SSM and the Startup Daemons must be started in a different way. This section covers how to start the Startup Daemons, how to start SSM, and how to start the other HPSS servers once SSM and the Startup Daemons are up and running.

16.6.2.1. Starting the Startup Daemons

Invoke the `/opt/hpss/bin/rc.hpss` script as root to start the HPSS Startup Daemon. By default, both stdout and stderr are redirected to the logfile `HPSS_PATH_LOG/hpss_stdout_stderr.log` and not to `/dev/console` since this can have a negative impact on performance. Specifying the **-o** option will cause redirection of output to `/dev/console`.

To start the Startup Daemon, use the **-d** option to rc.hpss(7):

```
% su -
```

```
% /opt/hpss/bin/rc.hpss -d [start]
```

16.6.2.2. Starting SSM

The SSM System Manager configuration metadata should have already been created by **mkhpss** as part of the infrastructure configuration. After SSM is started, this configuration metadata may be modified if necessary. Refer to the HPSS infrastructure for more information.

The SSM server startup script, rc.hpss(7), can be used to invoke the HPSS System Manager. The server requires a number of environment variable values to be used as its command line arguments and to be used to configure and manage the HPSS servers. These variables are defined by the system. The system defined values can be overridden in the `env.conf` file. They should have been edited to reflect the site's configuration during the HPSS infrastructure configuration phase.

Before starting up the System Manager, ensure that DB2 and other prerequisite software are up and running.

The rc.hpss(7) script will not start the System Manager if it is already running on the same node. Ensure that only one instance of the SSM System Manager is running at any one time.

To start the SSM System Manager, use the **-m** option to rc.hpss(7):

```
% su -
% /opt/hpss/bin/rc.hpss -m [start]
```

If the SSM's server name has changed since mkhpss was run, the environment variable HPSS_DESC_SSMSM must be set to match the new server name.

For best performance, it is recommended that the SSM graphical user interface (GUI) be installed on the SSM user's desktop machine. However, authorized SSM users can invoke the hpssgui(7) script directly from the HPSS server machine if desired. However, most users of the SSM command line interface (hpssadm(7)) will most likely invoke the script on the server machine.

16.6.2.3. Starting other HPSS servers

Other servers are started using SSM or can be automatically started upon startup of the System Manager by using the Core Server **Auto Startup** feature. This feature can be set using the *Core Server Configuration* window. Refer to HPSS infrastructure.

When starting a server from SSM using the *Servers* window, one or more HPSS servers can be started by the same request. SSM forwards the server's configuration data to the appropriate HPSS Startup Daemon. The Startup Daemon invokes the server with the appropriate input arguments. After the server is started, SSM attempts to establish a connection with the server. The server status is reported in the **Status** field on the *Servers* window. Refer to Server list for more information.

Before starting a server, ensure that DB2 and other prerequisite software are running, Kerberos is running (if using Kerberos authentication), that the Startup Daemon on the server's node is running and has a connection with SSM, and that the server is configured and its **Executable** flag is set. Ensure that any required PVR controllers are up and running before bringing up the PVRs.

To start a server, select one or more desired servers from the *Servers* window and click on the **Start** button. Verify the result of the request in the message area on the *Servers* window. In addition, monitor the *Alarms and Events* window for the "Server Initialized" event.

The Startup Daemon allows only one instance of a configured server to be brought up at a time. If a server is already running, the subsequent startup request for that server will fail.

16.6.2.4. Automatic server restart

HPSS servers may be configured to automatically restart. An **Auto Restart Count** is contained in each server configuration. This count may be used to define the number of times a failed server will be automatically restarted without any administrative action. The **Auto Restart Count** is ignored for the Startup Daemon and the SSM System Manager.

A failure is defined as a server terminating abnormally, and an abnormal termination is the result of the server exiting with a nonzero return code or as a result of a signal other than SIGTERM or SIGKILL. Termination due to an administrative halt or shutdown is not considered to be an abnormal termination and a server will not be automatically restarted.

The following **Auto Restart Count** values may be set:

Table 21. Auto Restart Count values

Auto Restart Count value	Meaning
0	Do not restart
-1	Infinite restart attempts
n	Restart a failed server n times (where n > 0)

After the **Auto Restart Count** is exhausted, the Startup Daemon will not attempt to restart the server. If the server executes for more than one hour without failing, the number of failures will be set back to zero.



*Do not confuse this feature with the Core Server **Auto Startup** feature. This feature can be set using the Core Server Configuration window described in Execution controls.*

16.7. Stopping HPSS

16.7.1. Shutting down an HPSS server

One or more HPSS servers can be shut down in a single SSM request. SSM will send each server a request to perform a controlled shutdown. The server will attempt to complete its current tasks, discontinue accepting new tasks, and inform SSM of its termination prior to exiting.

To initiate the request, select the desired servers on the *Servers* window and click on the **Shut Down** button. SSM will confirm the request through a pop-up window. The SSM user may wish to

verify the result of the request in the status bar on the *Servers* window as well as to monitor the *Alarms and Events* window for the "Server Terminated" event.



*HPSS Startup Daemons and the SSM System Manager cannot be shut down from the Servers window. Select **System Manager** from the **Shutdown** submenu of the **Operations** menu of the HPSS Health and Status window to shut down the System Manager. Use the rc.hpss(7) script stop option to shut down either the System Manager or the Startup Daemon.*

Servers may not terminate immediately since they may wait for pending operations to complete before terminating. Monitor the *Alarms and Events* window for possible notifications of delay from the server. In addition, monitor the *Servers* window for the server's shutdown status. If an immediate shutdown is required, use the **Force Halt** operation described in Halting an HPSS server.

SSM must have a connection to the Startup Daemon associated with the server or the shutdown operation will fail.

16.7.2. Shutting down all HPSS servers

Most HPSS servers (all but SSM and the Startup Daemon) can be shut down by selecting **Shut Down All Non-SSM Servers** from the **Shutdown** submenu of the **Operations** menu. SSM will process this request similarly to the **Shutdown Server** request issued from the *Servers* window.

This feature is useful when the entire HPSS system must be shut down, leaving SSM up to monitor the server shutdown process. When the servers are down, the SSM server and sessions can be shut down by selecting **System Manager** from the **Shutdown** submenu of the **Operations** menu. The SSM client sessions will detect that the SSM System Manager has been shut down and will then inform the user of the event.

16.7.3. Halting an HPSS server

Halting an HPSS server stops it without giving it an opportunity to shut down gracefully. When the command is received by the server, the server simply terminates immediately without doing any of its normal shutdown processing. It should be used only if it is necessary to shut down the server as quickly as possible or if attempts to shut the server down using the **Shutdown** button have failed.

One or more HPSS servers can be halted in a single request. SSM will issue a request to each selected server's Startup Daemon to issue a SIGKILL signal to its HPSS server.

To initiate the request, select the desired servers on the *Servers* window and click on the **Force Halt** button. SSM will confirm the request through a pop-up window. Verify the result of the request in the status bar on the *Servers* window. In addition, monitor the *Alarms and Events* window for the "Server Halted" alarm.

HPSS Startup Daemons and the SSM System Manager cannot be halted from the *Servers* window.

SSM must have a connection to the Startup Daemon associated with the server or the Halt Server operation may fail.

16.7.4. Shutting down the SSM server

To shut down the SSM server, select **System Manager** from the **Shutdown** submenu of the **Operation** menu on the *HPSS Health and Status* window. All SSM user sessions, including graphical and command line sessions, will detect that the System Manager has exited. Choosing this option will pop up a confirmation window which allows the shutdown request to be approved or canceled.

As the System Manager exits, a notification window will pop up on each logged-on SSM graphical user interface session informing the user that the GUI has lost connection to the System Manager. The session user can click the **Exit** button on this window to exit the SSM session or the **Login** button to log out of the session and get a new login window, which can be used once the System Manager has been restarted.

At each command line user interface session, a notification message will be displayed that the client connection to the System Manager has been lost and the session will be terminated.

These notifications will also happen whenever the client is having communication problems with the System Manager, even if the System Manager is still running.

Additionally the System Manager can be shut down with the **-m** option of rc.hpss(7):

```
% su -
% /opt/hpss/bin/rc.hpss -m stop
```

16.7.5. Shutting down the Startup Daemon

To stop the Startup Daemon, use the **-d** option of rc.hpss(7):

```
% su -
% /opt/hpss/bin/rc.hpss -d stop
```

16.7.6. Stopping the prerequisite software

To stop the prerequisite software:, use the **-p** option of rc.hpss(7):

```
% su -
% /opt/hpss/bin/rc.hpss -p stop
```

This will stop Kerberos, LDAP and DB2 if the corresponding shutdown scripts exist in `/var/hpss/etc`.

16.8. Server repair and reinitialization

This section describes how to repair or reinitialize an HPSS server.

16.8.1. Repairing an HPSS server

Repairing a server is the process of instructing the server to reset its internal state and status variables to their nominal settings. These variables include Operational State, Usage State, Administrative State, Service Status, Security Status, Software Status, Hardware Status, and Communication Status. Operational State is displayed prominently on the *Servers* window. If a server encounters an error as it runs, it may set one or more of these state and status variables to a value indicating the type and severity of the error. Alarm messages may be generated as well. Repairing a server simply resets all of these state and status variables to their nominal settings. One or more servers can be repaired with each request.

To initiate the request, select the desired servers from the *Servers* window and click on the **Repair** button. Verify the result of the request in the status bar on the *Servers* window. In addition, monitor the *Alarms and Events* window for the "Server Repaired" event.

Repairing a server does not correct the underlying problem that caused the server's reported state to change. Rather, it is a means for the administrator to notify the server that the underlying problem has been corrected or dismissed. It is an "alarm reset".

16.8.2. Reinitializing a server

Reinitializing a server causes it to reread all or part of its configuration metadata and reset its internal representation of those values. In many cases reinitialization or clicking the **Apply Config** button is all that is needed to cause a server to start using a new configuration. In other cases it may be necessary to restart a server to cause it to use a new configuration.

HPSS servers vary widely in their support of the reinitialization function. The following table lists the behavior of each HPSS server when it is told to reinitialize.

Table 22. Server reinitialization behavior

Server	Behavior
Core Server	• Resets the **Maximum Active I/O Requests**, **Maximum Active Copy Requests**, **COS Copy to Disk**, **COS Change Retry Count**, **Tape Dismount Delay**, **Tape Handoff Delay**, **PVL Max Connection Wait** and **Fragment Smallest Block** values to the values in the specific configuration metadata record. Note that reducing the Maximum Active I/O Requests or Maximum Active Copy Requests does not result in any running operations being quiesced - use I/O Abort to reduce the total number of running operations. • Sets the server's diagnostic Logging Options to the values in the specific configuration metadata record. • Reloads cached Class of Service information for the system. • Reloads cached **Migration Policy** for those policies that were already cached in the server's memory. Does not add new policies to, or remove deleted policies from the cache. • Reloads cached global **Root User ID**, **Root is Superuser**, **Can change UID to self if has Control Perm**, **Can change UID if has Delete Perm on Security ACL**, **Object names can contain unprintable characters**, **Trashcans Enabled**, **Number of Trashcan Threads**, **Trash Incinerator Interval**, **Trash Unlink Eligible Time**, and **Trash Statistics Interval** values using values read from the global configuration metadata. • Reloads cached global and subsystem **Metadata Space Warning Threshold**, **Metadata Space Critical Threshold**, **Metadata Space Monitor Interval**, **DB Log Monitor Interval**, and **COS Change Stream Count** values using values read from the global and subsystem configuration metadata. • Reloads cached **Default Class of Service** and **Gatekeeper** values using values read from the subsystem configuration metadata. • Releases the in-memory cache of File Family IDs, allowing it to be reconstructed. If a file family has been removed from the list of defined families, this step causes it to be unknown to the Core Server. • Reinitializes the in-memory cache of environment variable values - rereads the `env.conf` file.

Server	Behavior
Core Server (continued)	• Adjusts the maximum number of open files supported by the Core Server when the number of migration streams increases. • Refreshes the Core Server's list of available RAIT Engines. • Refreshes the Core Server's Manage Object with the new values from this list. • Refreshes the Core Server's cached Mover and RAIT Engine configuration metadata. If a Mover or RAIT Engine's configuration metadata is changed, reinitializing the Core Server will make the Core Server aware of the changes. • Closes all unused disk Mover communication sockets. • Rereads the accounting policy and refreshes the account validation cache. • Updates server basic configuration thread, request queue, and connection configuration. Any Core Server configuration information not listed above is not changed when the server is reinitialized. For example, security-related configuration settings.
SSM	This server rereads the environment. Update server basic configuration thread, request queue, and connection configuration.
Mover	Rereads log policy only. Update server basic configuration thread, request queue, and connection configuration.
PVR	Reinitializes the internal drive list cache. For the SCSI PVR, also reinitialize the control path list. Update server basic configuration thread, request queue, and connection configuration.
MPS	Reloads the **Storage Class Update Interval**, **Core Server API Failures Allowed**, and **MPS Report File Name** type-specific configuration values.
GK	Reloads the *Accounting Policy* **Options** flags.
All other servers	Update server basic configuration thread, request queue, and connection configuration.

To reinitialize a server, select the desired servers from the *Servers* window and click on the **Reinitialize** button. SSM will confirm the request through a pop-up window. The request results may be verified via the status bar on the *Servers* window as well through the *Alarms and Events* window.



If a configuration setting has been changed and that change can be applied through reinitialization, the **Apply Config** button on the *HPSS Health and Status* screen will become enabled. Clicking the **Apply Config** button will reinitialize all of the servers, except the Startup Daemon and the System Manager, and it is the recommended method to ensure the setting is applied.

If a server does not support reinitialization, a pop-up window will be displayed to inform the user of the invalid request. Warnings are not displayed for servers that only partially support reinitialization. Servers that do not support reinitialization, or those that do not support reinitializing the settings in question, must be restarted in order for configuration modifications to take effect.

Some groups of servers depend on consistent configuration information to run properly. For example, the Core Server and Migration/Purge Server must agree on Class of Service, hierarchy, and storage class configurations. When changes are made to these sorts of configuration information, all of the servers that rely on these settings should be restarted to make sure they agree. In most cases, as mentioned in the note above, the **Apply Config** button can be used and is the recommended method to ensure these configuration settings are applied across all of the servers.

16.9. Forcing an SSM connection

When the status of the server's connection with SSM is UP/UNCONNECTED, select the server and then click on the **Force Connect** button on the *Servers* window. SSM will attempt to establish the connection with the server and reflect the latest data in the **Status** field on the *Servers* window.

The SSM System Manager periodically retries the connection with all the servers that are configured to execute. The **Force Connect** button may be used to trigger an immediate connection request.

Chapter 17. Storage configuration

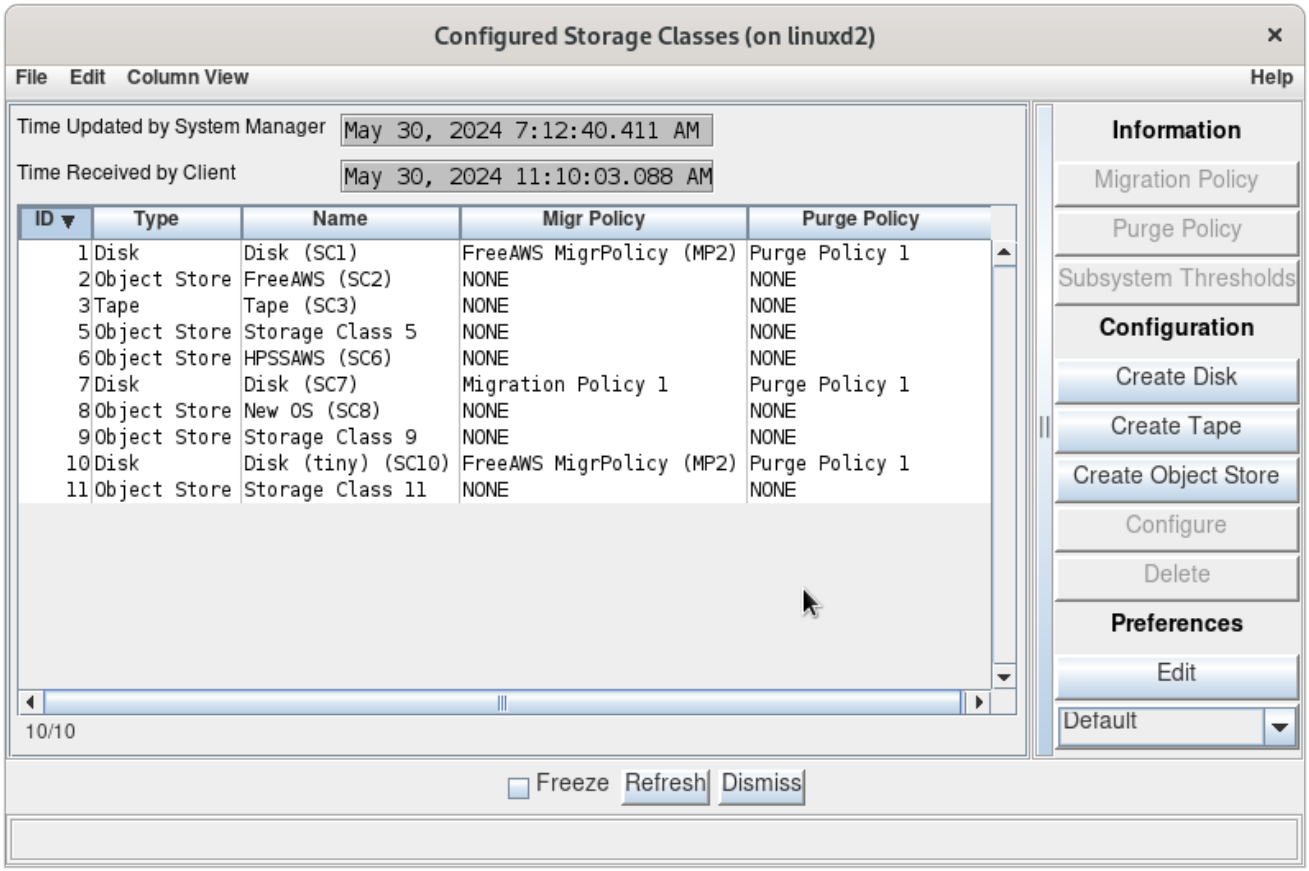
This chapter describes the procedures for creating, modifying, and deleting storage classes, hierarchies, Classes of Service, migration policies, purge policies, and file families.

17.1. Storage classes

This section describes the process of configuring storage classes.

17.1.1. Configured Storage Classes window

A storage class can be created and managed using the *Configured Storage Classes* window.



This window displays all storage classes in the HPSS system, both empty and non-empty. This window is displayed by first selecting **Configure** from the *HPSS Health and Status* window. This will display a drop-down menu. Select **Storage Space** from this menu. This will display another drop-down menu. Select **Storage Classes** from this menu.



When adding or changing a storage class, the Core and Migration/Purge Servers must be reinitialized or restarted in order for the change to take effect. It is recommended to use the **Apply Config** button on the *HPSS Health and Status* window to reinitialize the servers.

See also the related window *Active Storage Classes*, as described in Active Storage Classes window.

The *Active Storage Classes* window does not list any empty storage classes, but only those to which volumes have been assigned.

Field descriptions

Storage Classes List

The central panel in this window shows the list of configured storage classes. The columns display selected fields from the storage class configuration windows discussed in the following sections.

Information buttons

Migration Policy

Opens the configuration window for the migration policy that is configured for the selected storage class. This button will be disabled if no storage classes are selected in the storage classes list or the selected storage class does not have a migration policy.

Purge Policy

Opens the configuration window for the purge policy that is configured for the selected storage class. This button will be disabled if no storage classes are selected in the storage classes list or the selected storage class does not have a purge policy.

Subsystem Thresholds

Opens either a disk or tape subsystem threshold configuration window, as appropriate, for the selected storage class.

Configuration buttons

Create Disk

Opens a *Disk Storage Class Configuration* window containing default values for a new disk storage class.

Create Tape

Opens a *Tape Storage Class Configuration* window containing default values for a new tape storage class.

Create Object Store

Opens a *Object Store Storage Class Configuration* window containing default values for a new object store storage class.

Configure

Opens the selected storage class configurations for editing.

Delete

Deletes the one or more selected storage classes.

17.1.2. S3 Polling Intervals

It is recommended to configure the restore notifications if possible within the environment. This will reduce the wait time between when the restore completes and when HPSS detects that the restore has been completed.

However, there are some environments where this may not be possible. In these situations, HPSS will poll (via HEAD request) the AWS system waiting for the specified object to be restored. Polling will start at the minimum time of object retrieval, and continue until the maximum time of object retrieval. After the maximum time it will poll at a fixed number of times at a reduced interval, before finally giving up and declaring the recall a failure.

Each type of object store has different polling intervals. The following tables indicate the polling intervals for each type.

Table 23. S3 Glacier Deep Archive

Recall Priority	Start Poll	Poll Stop	Polling Interval	Final Polls	Final Polling Interval
Bulk	48 hours	N/A	48 hours	3	5 minutes
Standard	12 hours	N/A	12 hours	3	5 minutes

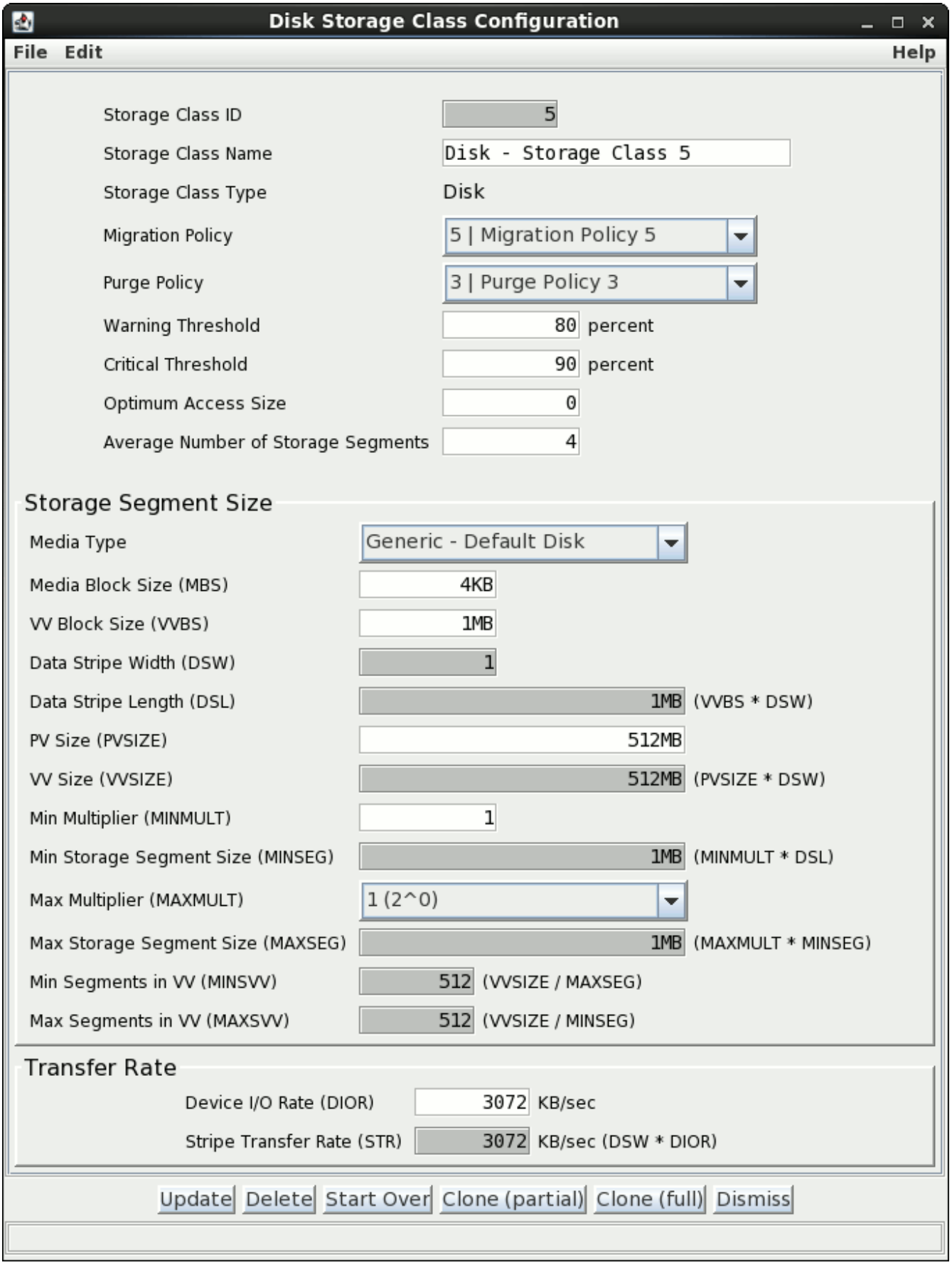
Table 24. S3 Glacier Flexible Retrieval

Recall Priority	Start Poll	Polling Interval	Poll Stop	Final Polls	Final Polling Interval
Bulk	5 hours	1 minute	12 hours	3	5 minutes
Standard	3 hours	1 minute	5 hours	3	5 minutes
Expedited	1 minute	30 seconds	5 minutes	5	1 minute

Table 25. S3 Intelligent Tiering

Location	Recall Priority	Start Poll	Polling Interval	Poll Stop	Final Polls	Final Polling Interval
Archive	Bulk	5 hours	1 minute	12 hours	3	5 minutes
Archive	Standard	3 hours	1 minute	5 hours	3	5 minutes
Archive	Expedited	1 minute	20 seconds	5 minutes	5	1 minute
Deep Archive	Bulk	48 hours	N/A	48 hours	3	5 minutes
Deep Archive	Standard	12 hours	N/A	12 hours	3	5 minutes

17.1.3. Disk Storage Class Configuration



This window is used to manage disk storage class configurations.

Field descriptions

Storage Class ID

The numeric identifier assigned to the storage class.

Storage Class Name

The descriptive name of the storage class.

Storage Class Type

The type of the storage class (always `Disk`).

Migration Policy

The migration policy associated with this storage class, or `None` if no migration is desired.



Advice

Do not configure a migration policy for a storage class at the lowest level in a hierarchy.

If a migration policy is added to a storage class after files are created in the storage class, those files may never be migrated. Use the mkmprec(7) utility to correct this problem.

Purge Policy

The purge policy associated with this storage class, or `None` if no purge is desired.



Advice

Do not configure a purge policy for a tape storage class or for any storage class which does not have a migration policy in effect. Purge policies only apply to disk storage classes. Purging from tape is managed as a special characteristic of the tape migration policy.

Warning Threshold

A threshold for space used in this storage class expressed as a percentage of the total space defined in the storage class. Alarms will be sent to SSM periodically when free space in the storage class falls below this value, and the **Space Thresholds** field on the *HPSS Health and Status* window will be changed to `Warning`. Note that this field will not have any effect if overridden by storage subsystem-specific thresholds.

Critical Threshold

Another threshold for space used in this storage class expressed as a percentage of the total space defined in the storage class. Alarms will be sent to SSM periodically when free space in the storage class falls below this value, and the **Space Thresholds** field on the *HPSS Health and Status* window will be changed to `Critical`. Note that this field will not have any effect if overridden by storage subsystem-specific thresholds.



Advice

These values determine limits beyond which the MPS sends alarms indicating that storage space is running low. The thresholds on disk are measured in bytes and expressed in terms of the percentage of total space in the storage class. The threshold

may be disabled by setting the value to 100 percent.

Optimum Access Size

The optimal transmission size to be used for a transfer request using this storage class. (Not currently used by HPSS. May be used by site-specific applications.)

Average Number of Storage Segments

The **Average Number of Storage Segments** is used by the Core Server as an aid in picking the size of disk storage segments when writing a file to disk. This field is ignored when the storage class is at the top level of the hierarchy in a COS which uses Variable Length or Fixed Length Max Style segment allocation. See the discussion of the COS **Allocation Method** field and **Truncate Final Segment** flag in Class of Service Configuration window for details on segment allocation methods.

When the storage class is at the top level of a COS which uses Fixed Length Classic Style segment allocation, or when the storage class is at any level below the top, regardless of the COS segment allocation method, the Core Server requires that all of the disk storage segments occupied by a given file be the same size, and must be within the range of sizes given by the Minimum Storage Segment Size and Maximum Storage Segment Size parameters.

If **Max Storage Segment Size (MAXSEG)** is greater than **Min Storage Segment Size (MINSEG)**, the Core Server attempts to select a storage segment size for a file such that the file can be stored in **Average Number of Storage Segments** segments. If the **Max Storage Segment Size (MAXSEG)** is equal to the **Min Storage Segment Size (MINSEG)**, the Core Server will use this value for the storage segment size, and the file will occupy as many of these segments as necessary.

The principal effect of changing the value of the **Average Number of Storage Segments** is on disk volume fragmentation. If the files to be stored in a storage class are large relative to the size of the disk VVs, fragmentation of the volumes may make it difficult to find space for a new segment. Setting **Average Number of Storage Segments** to a larger value will increase the number of segments occupied by files, and decrease the segment size. Fragmentation of the volumes will be reduced, but the amount of metadata required to describe the files will be increased.

Conversely, if files are small relative to the size of the disk VVs, smaller values of **Average Number of Storage Segments** increase the size of the storage segments, and decrease the number of segments occupied by each file. This reduces the metadata storage requirements of the file.

The number of segments in small HPSS files can have a significant impact on transfer performance. To maximize the transfer performance, set this value to "1". Keep in mind that this will result in less effective disk space utilization. On average, you will use only 50% of your disk space with this selection.

Storage Segment Size

Media Type

The media type of all volumes in the storage class.

Media Block Size (MBS)

The **Media Block Size** is the size in bytes of a physical data block on the media. For disk, the value must be a multiple of the physical block size used by the disk hardware. For example, if

the disk hardware stores data in 512-byte sectors, 2048 would be a valid entry in this field, but 2000 would not.



Advice

*The **Media Block Size** should be set to a value appropriate for the volume type. See the Media block size selection section for some recommendations.*

VV Block Size (VVBS)

The virtual volume block size is the number of bytes written to an element of a striped VV before the data stream moves to the next stripe element. It can be thought of as the stride length of striped data transfer operations. The length of the VV block has an effect on the striping efficiency. Short VV blocks cause more protocol overhead when writing striped devices. In non-striped applications, VV Block Size has little meaning so any convenient multiple of the Media Block Size will do.



Advice

When choosing a VV Block Size, the administrator should consider the characteristics of any data source or sink that will be copied to or from. Best performance of striped copies usually occurs when the VV Block Sizes of the source and sink are equal. This minimizes the data movement protocol overhead and helps to keep the data streams flowing smoothly.

VV Block Size must be an integer multiple of the Media Block Size.

See the Virtual volume block size selection (disk) and Virtual volume block size selection (tape) sections for information.

Data Stripe Width (DSW)

The number of physical volumes in a virtual volume in this storage class.

Data Stripe Length (DSL)

The **Data Stripe Length (DSL)** is the product of the **VV Block Size (VVBS)** and the **Data Stripe Width (DSW)**.

PV Size (PVSIZE)

The size of the disk volume in bytes.

VV Size (VVSIZE)

The virtual volume size. The product of the **PV Size (PVSIZE)** and the **Data Stripe Width (DSW).**

Min Multiplier (MINMULT)

The **Min Storage Segment Size (MINSEG)** field is set to the product of this value and **Stripe Length (SL)**.

Min Storage Segment Size (MINSEG)

The lower bound for storage segment sizes created on volumes in this storage class. This

value is the product of the **Stripe Length (SL)** and the **Min Multiplier (MINMULT)**.

Max Multiplier (MAXMULT)

The **Max Storage Segment Size (MAXSEG)** must be a power of two multiple of the **Stripe Length (SL)**. This selection list contains the valid power of two values from 1 (2^0) to 16,777,216 (2^{24}). Select the appropriate multiplier from the selection list. The field **Max Storage Segment Size (MAXSEG)** is set to the product of this value and **Min Storage Segment Size (MINSEG)**.

Max Storage Segment Size (MAXSEG)

The upper bound for storage segment sizes created on volumes in this storage class. This must be a power of two multiple of the **Stripe Length (SL)**. This value is the product of the **Min Storage Segment Size (MINSEG)** and the **Max Multiplier (MAXMULT)**.

Min Segments in VV (MINSVV)

The minimum number of segments in a virtual volume. This value is calculated as the **VV SIZE (VVSIZE)** parameter divided by the **Max Storage Segment Size (MAXSEG)** parameter.

Max Segments in VV (MAXSVV)

The maximum number of segments in a virtual volume. This value is calculated as the **VV SIZE (VVSIZE)** parameter divided by the **Min Storage Segment Size (MINSEG)** parameter.

Transfer Rate

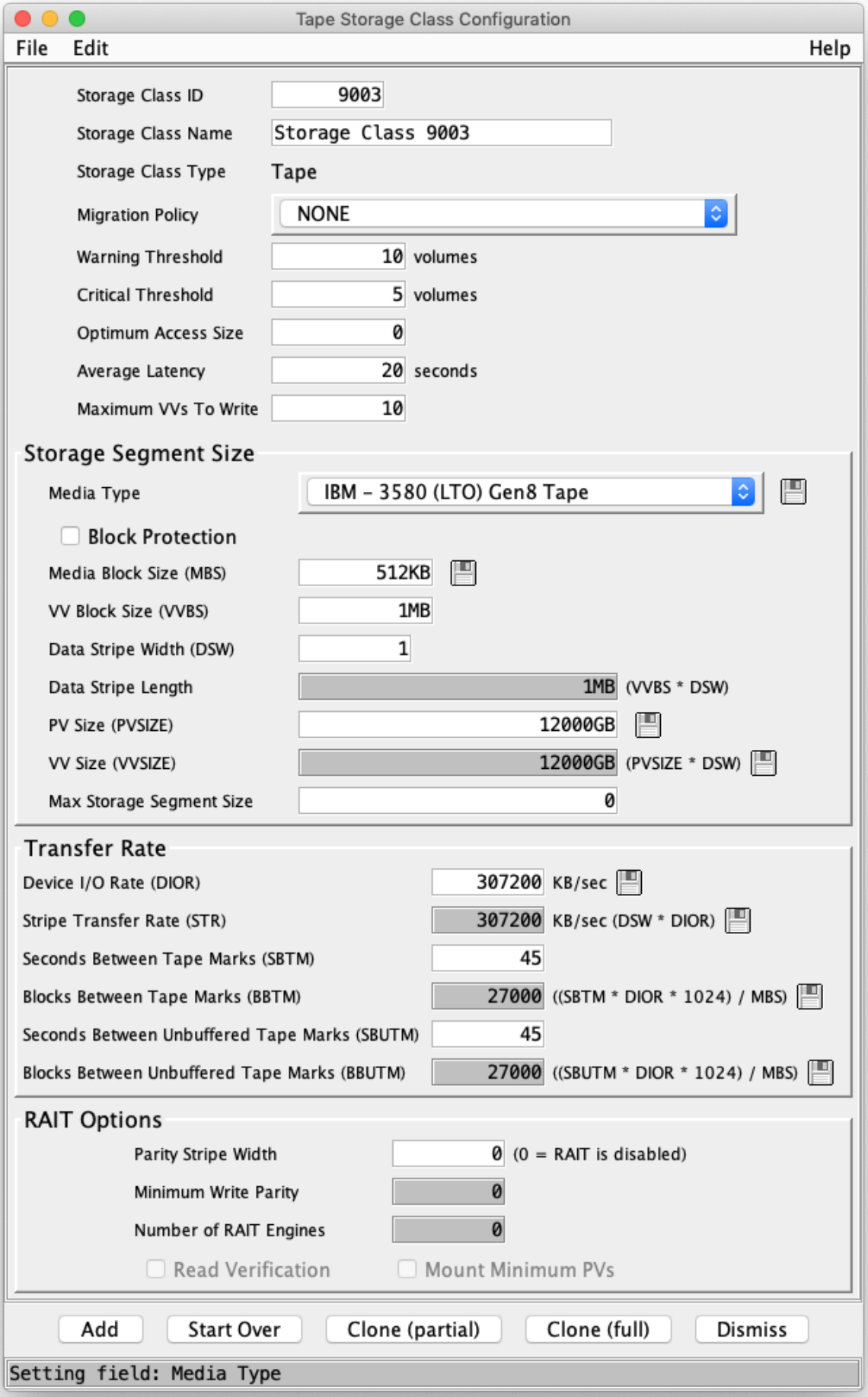
Device I/O Rate (DIOR)

The approximate data transfer speed, in kilobytes per second (KB/sec), which can be achieved by devices corresponding to **Media Type**. In disk storage classes, this field is informational only.

Stripe Transfer Rate (STR)

The estimated aggregate transfer rate for volumes of this type. This value is the product of the **Data Stripe Width (DSW)** and the **Device I/O Rate (DIOR)**.

17.1.4. Tape Storage Class Configuration


Tape Storage Class Configuration
File
Edit
Help
Storage Class ID
9003
Storage Class Name
Storage Class 9003
Storage Class Type
Tape
Migration Policy
NONE
Warning Threshold
10
volumes
Critical Threshold
5
volumes
Optimum Access Size
0
Average Latency
20
seconds
Maximum VVs To Write
10
Storage Segment Size
Media Type
IBM – 3580 (LTO) Gen8 Tape
Block Protection
Media Block Size (MBS)
512KB
VV Block Size (VVBS)
1MB
Data Stripe Width (DSW)
1
Data Stripe Length
1MB
(VVBS * DSW)
PV Size (PVSIZE)
12000GB
VV Size (VVSIZE)
12000GB
(PVSIZE * DSW)
Max Storage Segment Size
0
Transfer Rate
Device I/O Rate (DIOR)
307200
KB/sec
Stripe Transfer Rate (STR)
307200
KB/sec (DSW * DIOR)
Seconds Between Tape Marks (SBTM)
45
Blocks Between Tape Marks (BBTM)
27000
((SBTM * DIOR * 1024) / MBS)
Seconds Between Unbuffered Tape Marks (SBUTM)
45
Blocks Between Unbuffered Tape Marks (BBUTM)
27000
((SBUTM * DIOR * 1024) / MBS)
RAIT Options
Parity Stripe Width
0
(0 = RAIT is disabled)
Minimum Write Parity
0
Number of RAIT Engines
0
Read Verification
Mount Minimum PVs
Add
Start Over
Clone (partial)
Clone (full)
Dismiss
Setting field: Media Type

This window is used to manage tape storage class configurations.

Field descriptions

Storage Class ID

The numeric identifier assigned to the storage class.

Storage Class Name

The descriptive name of the storage class.

Storage Class Type

The type of the storage class (always `Tape`).

Migration Policy

The migration policy associated with this storage class, or `None` if no migration is desired.



Advice

Do not configure a migration policy for a storage class at the lowest level in a hierarchy.

If a migration policy is added to a storage class after files are created in the storage class, those files may never be migrated. Use the mkmprec(7) utility to correct this problem.

Warning Threshold

A threshold for space used in this storage class expressed as a number of empty tape volumes. Alarms will be sent to SSM periodically when the number of empty tapes in the storage class falls below this value, and the **Space Thresholds** field on the *HPSS Health and Status* window will be changed to `Warning`. Note that this field will not have any effect if overridden by storage subsystem-specific thresholds.

Critical Threshold

Another threshold for space used in this storage class expressed as a number of empty tape volumes. Alarms will be sent to SSM periodically when the number of empty tapes in the storage class falls below this value, and the **Space Thresholds** field on the *HPSS Health and Status* window will be changed to `Critical`. Note that this field will not have any effect if overridden by storage subsystem-specific thresholds.

Optimum Access Size

The optimal transmission size to be used for a transfer request using this storage class. (Not currently used by HPSS. May be used by site-specific applications.)

Average Latency

The average time (in seconds) that elapses when a data transfer request is scheduled and the time the data transfer begins.

Maximum VVs to Write

The number of tape virtual volumes in the storage class that a Core Server will use for

concurrent writes.



Advice

Small values in this field restrict files being written in the storage class to a small number of tapes, reducing the number of tape mounts. The number of tape drives used to write files in the storage class will be limited to approximately the value of this field times the total stripe width of the mounted VVs. Read operations are not limited by this value.

Storage Segment Size

Media Type

The media type associated with the storage class.

Block Protection

When this flag is set, Logical Block Protect (LBP) will be enabled for tape volumes created in this storage class. When LBP is enabled, HPSS will generate and pass a CRC with each block written to the tape volume and validate the CRC accompanying each block read from the tape volume. This function is supported for the following media types.

- LTO (Generation 5 and greater)
- IBM 3592 (E07 and greater)
- Oracle T10000 (C and greater)

This option is not supported by virtual tape devices such as AWS Tape Gateway and should be disabled.

Media Block Size (MBS)

The Media Block Size is the size in bytes of a physical data block on the media. For tape, this can be almost any value within reasonable limits. If the tape hardware has a recommended physical block size, use that value.



Advice

*The **Media Block Size** should be set to a value appropriate for the volume type. See the Media block size selection section for some recommendations.*

See also Device I/O Rate and Seconds Between Tape Marks in this section.

VV Block Size (VVBS)

The virtual volume block size is the number of bytes written to an element of a striped VV before the data stream moves to the next stripe element. It can be thought of as the stride length of striped data transfer operations. The length of the VV block has an effect on the striping efficiency. Short VV blocks cause more protocol overhead when writing striped devices. In non-striped applications, **VV Block Size** has little meaning so any convenient multiple of the **Media Block Size** will do.



Advice

When choosing a VV Block Size, the administrator should consider the characteristics of any data source or sink the storage class that will be copied to or from. Best performance of striped copies usually occurs when the VV Block Sizes of the source and sink are equal. This minimizes the data movement protocol overhead and helps to keep the data streams flowing smoothly.

VV Block Size must meet the following constraining requirements:

- *It must be an integer multiple of the Media Block Size.*
- *The PV Section Length (**Media Block Size (MBS)** multiplied by the **Blocks Between Tape Marks (BBTM)**) divided by the **VV Block Size (VVBS)** must be a whole number. For example, if the **Media Block Size (MBS)** is 64 KB, and the **Blocks Between Tape Marks (BBTM)** is 512, the physical volume section length is 32 MB. The **VV Block Size (VVBS)** could be 64 KB, 128 KB, 256 KB, or larger, but not 192 KB.*

See the Virtual volume block size selection (disk) and Virtual volume block size selection (tape) for information.

Data Stripe Width (DSW)

The number of physical volumes in a virtual volume in this storage class. In the case of RAIT VVs, this is the number of PVs that contain file data.

Data Stripe Length

The **Data Stripe Length** is the product of the **VV Block Size (VVBS)** and the **Data Stripe Width (DSW)**.

PV Size (PVSIZE)

The estimated size of the tape volume in bytes. The actual number of bytes that can be written to a tape will vary with individual media and data compressibility. Think of this field as the nominal length of a tape of this type.

VV SIZE (VVSIZE)

The virtual volume size. The product of the **PV Size (PVSIZE)** and the **Data Stripe Width (DSW)**.

Max Storage Segment Size

The upper bound for storage segment sizes created on volumes in this storage class. A value of "0" indicates that the maximum segment size is unlimited. If the value entered is nonzero, then it must be between 10% and 50% of the VV estimated size (**VV Size (VVSIZE)**).

Transfer Rate

Device I/O Rate (DIOR)

The approximate data transfer speed, in kilobytes per second (KB/sec), which can be achieved by devices corresponding to Media Type.



Advice

*This field is used in calculating a reasonable setting for **Seconds Between Tape Marks (SBTM)**.*

Stripe Transfer Rate (STR)

The estimated aggregate transfer rate for volumes of this type. This field is calculated as the **Data Stripe Width (DSW)** multiplied by the **Device I/O Rate (DIOR)**.

Seconds Between Tape Marks (SBTM)

The number of seconds between tape mark writes while writing continuously to tape.

Advice

*The **Device I/O Rate (DIOR)**, the **Media Block Size (MBS)** and the **Seconds Between Tape Marks (SBTM)** determine the length of a physical volume section. The administrator should set the **Device I/O Rate (DIOR)** to an appropriate value for the device, and then pick a **Media Block Size (MBS)**. Finally, adjust the **Seconds Between Tape Marks (SBTM)** to strike a balance between tape utilization and transfer efficiency.*



*Smaller values of **Seconds Between Tape Marks (SBTM)** cause more tape to be consumed in tape marks and break up data streaming throughput on streaming tape drives but cause controller buffers to be flushed more frequently and may improve access times to the middle of large files. Larger values improve transfer efficiency. However, smaller values may reduce the amount of data that may need to be rewritten on another tape when a tape reaches EOM. As tape drive streaming rates increase, we generally recommend larger values for this setting.*

Blocks Between Tape Marks (BBTM)

The maximum number of data blocks (of **Media Block Size (MBS)**) that will be written on a tape between consecutive tape marks.

The number of media data blocks between tape marks is calculated by SSM when the **Device I/O Rate (DIOR)** and **Seconds Between Tape Marks (SBTM)** are set. This value is displayed for information only.

The recording space between tape marks is called a "section". The PV Section Length is **Blocks Between Tape Marks (BBTM)** times the **Media Block Size (MBS)**. The VV Section Length is the PV Section Length multiplied by the **Data Stripe Width (DSW)**.

Seconds Between Unbuffered Tape Marks (SBUTM)

The number of seconds between synchronizing tape mark writes.

Advice



*When this value differs from **Seconds Between Tape Marks (SBTM)**, the Tape Mover is instructed to write buffered tape marks. A buffered tape mark will be written every SBTM seconds (or BBTM blocks), and an unbuffered/synchronizing tape mark will be written every SBUTM seconds (or BBUTM blocks).*

> *When buffered tape marks are enabled, write throughput is improved, but at the cost of potentially needing to rewrite more data in EOM cases. This is due to the behavior that when buffered tape marks are used, on an EOM or I/O error, the tape drive rolls the data written to tape back to the last tape mark that successfully made it to tape.*

Blocks Between Unbuffered Tape Marks (BBUTM)

The maximum number of data blocks (of **Media Block Size (MBS)**) that will be written on a tape between synchronizing tape marks.

RAIT Options

Parity Stripe Width

When this value is greater than zero, the VV is a RAIT VV. This is the number of PVs in a VV that will carry parity information.

Minimum Write Parity

The minimum number of PVs that may carry parity. This value sets a limit on the number of PVs that may become unwritable in a volume before the VV becomes unwritable. For example, if a RAIT VV is configured with four data PVs and two parity PVs (DSW = 4, PSW = 2), then Minimum Write Parity may be set to "2" or "1". If the value is "2", then if one of the PVs becomes unwritable, the VV may no longer be written. If MWP is set to "1", then if one PV becomes unwritable, the VV remains writable, but if two become unwritable, the VV becomes unwritable.

Minimum Write Parity may also be set to zero. In that case any data written to the VV after all of the parity PVs have become unwritable is not protected with parity. Any additional losses of PVs may render the data written this way unrecoverable.

Number of RAIT Engines

The number of RAIT Engines that will be used to read and write VVs in this storage class. The system has the ability to use multiple RAIT Engines in a striped I/O fashion to increase throughput.



> *Advice*
>
> *Unless it can be shown that the RAIT Engine is a bottleneck in a particular configuration, set **Number of RAIT Engines** to one.*

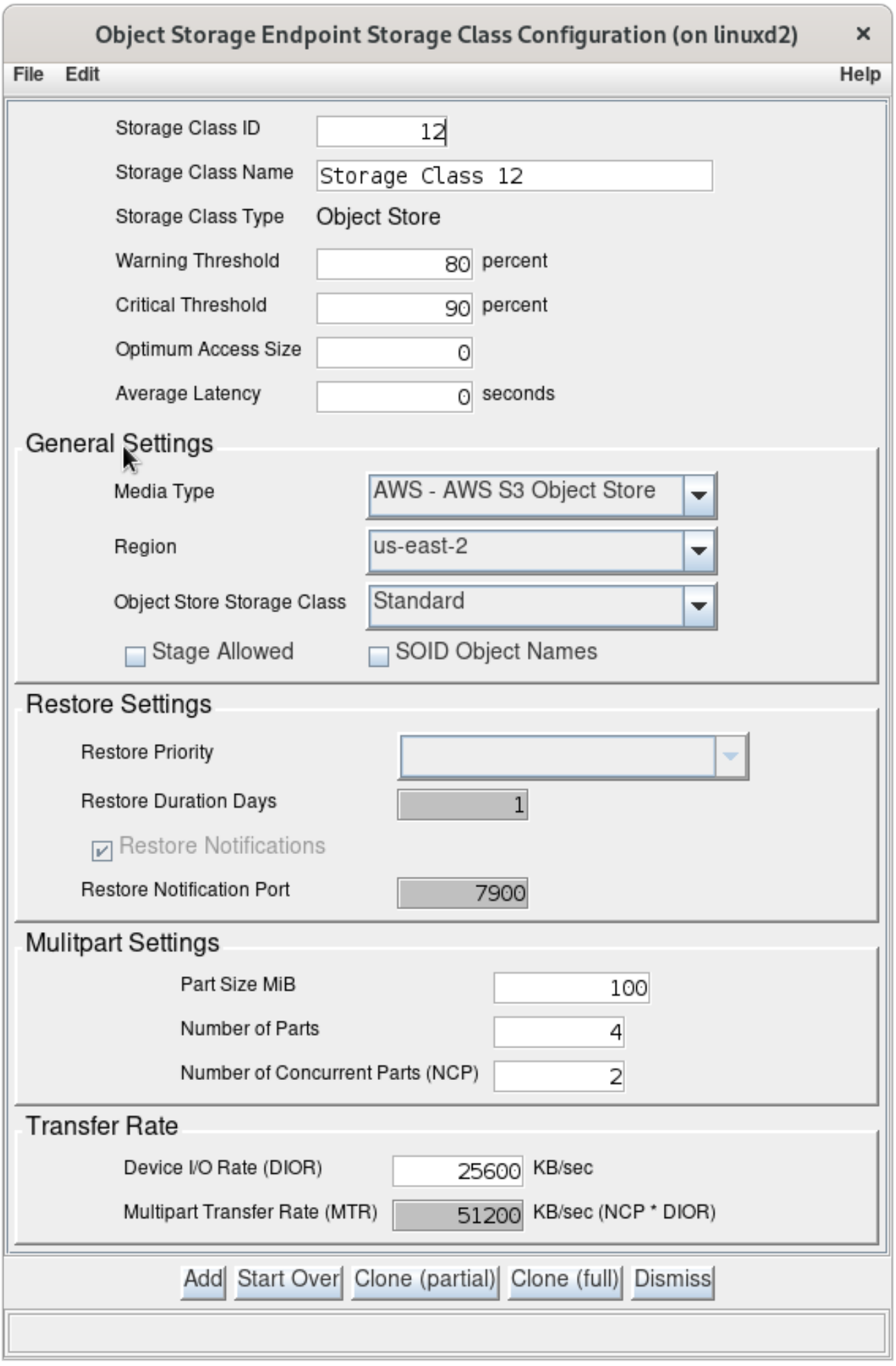
Read Verification

When selected, data written on VVs in the storage class will be verified using the parity information stored with the data. When not selected, the RAIT Engine reads the data from the VV but does not check it using the parity information.

Mount Minimum PVs

VVs created with this option will be read using the smallest number of PVs necessary to recover the data. If **Read Verification** is off, the number of PVs mounted will be equal to the value of DSW. If **Read Verification** is on, the number of PVs mounted will be equal to (DSW + 1).

17.1.5. Object Store Storage Class Configuration


Object Storage Endpoint Storage Class Configuration (on linuxd2)
File
Edit
Help
Storage Class ID
12
Storage Class Name
Storage Class 12
Storage Class Type
Object Store
Warning Threshold
80
percent
Critical Threshold
90
percent
Optimum Access Size
0
Average Latency
0
seconds
General Settings
Media Type
AWS - AWS S3 Object Store
Region
us-east-2
Object Store Storage Class
Standard
Stage Allowed
SOID Object Names
Restore Settings
Restore Priority
Restore Duration Days
1
Restore Notifications
Restore Notification Port
7900
Mulitpart Settings
Part Size MiB
100
Number of Parts
4
Number of Concurrent Parts (NCP)
2
Transfer Rate
Device I/O Rate (DIOR)
25600
KB/sec
Multipart Transfer Rate (MTR)
51200
KB/sec (NCP * DIOR)
Add
Start Over
Clone (partial)
Clone (full)
Dismiss

This window is used to manage object store storage class configurations.

Field descriptions

Storage Class ID

The numeric identifier assigned to the storage class.

Storage Class Name

The descriptive name of the storage class.

Storage Class Type

The type of the storage class (always `Object Store`).

Warning Threshold

A threshold for space used in this storage class expressed as a percentage of the total space defined in the storage class. Alarms will be sent to SSM periodically when free space in the storage class falls below this value, and the **Space Thresholds** field on the *HPSS Health and Status* window will be changed to `Warning`. Note that this field will not have any effect if overridden by storage subsystem-specific thresholds.

Critical Threshold

Another threshold for space used in this storage class expressed as a percentage of the total space defined in the storage class. Alarms will be sent to SSM periodically when free space in the storage class falls below this value, and the **Space Thresholds** field on the *HPSS Health and Status* window will be changed to `Critical`. Note that this field will not have any effect if overridden by storage subsystem-specific thresholds.

Optimum Access Size

The optimal transmission size to be used for a transfer request using this storage class. (Not currently used by HPSS. May be used by site-specific applications.)

Average Latency

The average time (in seconds) that elapses when a data transfer request is scheduled and the time the data transfer begins.

Media Type

The media type associated with the storage class.



Advice

*Currently, **AWS - AWS S3 Object Store** is the only supported object store media type. Once selected, other media type specific configuration settings will be available.*

Region

The region associated with the storage class.



Advice

AWS Regions are separate geographic areas that AWS uses to house its infrastructure. These are distributed around the world so that customers can choose a region closest to them in order to host their cloud infrastructure there. See the Amazon Simple Storage Service user guide for more information.

Object Store Storage Class

The AWS Storage Class used when storing data for this HPSS storage class.



Advice

Amazon S3 offers a range of storage classes that you can choose from based on the performance, data access, resiliency, and cost requirements of your workloads. S3 storage classes are purpose-built to provide the lowest cost storage for different access patterns. See the Amazon Simple Storage Service user guide for more information.

Stage Allowed

The flag that will allow stage requests to be issued for this storage class.



Advice

Additional costs are associated with accessing AWS data. Some sites utilizing AWS as a secondary copy, may prefer to restrict access to data stored in AWS to avoid unexpected costs. See the AWS Pricing Calculator for more information on costs associated with S3 storage.

SOID Names

When set, objects written to the endpoint will be named as their HPSS bitfile SOID value rather than their HPSS path name.



Advice

The default object naming behavior is to attempt to mirror the HPSS file path. However because object names have a 1024 byte size limitation and this is not always possible. When path mirroring is not possible the object name is generated from the corresponding bitfile SOID. This flag is used to change the default naming behavior to be based on the bitfile SOID.

Restore Priority

For archive type storage classes, this is the priority to be used when an object must be restored.



Advice

Expedited allows for quickly access to your data that is stored in the S3 Glacier Flexible Retrieval storage class or S3 Intelligent-Tiering Archive Access tier. **Standard** allows for access to your archived objects within several hours. **Bulk** provides the lowest-cost retrieval option, allowing retrieval of large amounts, even petabytes, of data inexpensively. See the AWS Pricing Calculator for more information on costs associated with S3 storage._

Restore Duration Days

The duration that restored data will remain available for recall.



Advice

There are additional costs associated with restored data. See the AWS Pricing Calculator for more information on costs associated with S3 storage.

Restore Notifications

This flags indicates that the restore requests will register for notification to be sent to HPSS at completion of a restore. If this flag is not set then the completion of restore is polled, using an interval based on the specified restore priority. See S3 Polling Intervals for more information.



Advice

AWS Simple Notification Service (SNS) is used to deliver the notifications. Additional AWS and SNS configuration is required to enabled notification delivery.

Restore Notification Port

This is the network port used for restore notifications.

Part Size MiB

This is the minimum number of megabytes used for each part of a multipart upload.



Advice

Valid values for the field are 5MiB to 5GiB.

Number of Parts

This is the number of parts for the upload.



Advice

The valid values for this field are 1 to 10,000. If this field is set to 1, multipart upload is disabled. If division of a file results in a part size (excluding the final part), less than the number of bytes specified in the previous parameter (i.e., "Part Size MiB") then the number of parts is reduced until the part size is greater than or equal to the configured part size.

Number of Concurrent Parts

This is the number of upload parts that are transmitted concurrently.



Advice

The valid values for this field are 1 to 10,000, where the value is less than to equal to the Number of Parts.

Device I/O Rate (DIOR)

The approximate data transfer speed, in kilobytes per second (KB/sec), which can be achieved by devices corresponding to Media Type.



Advice

This is the approximate AWS upload bandwidth for a single PUT. (This metric is currently used for informational purposes only).

Multipart Transfer Rate (MTR)

The maximum approximate data transfer speed, in kilobyte per second (KB/sec), which can be achieved for a multipart PUT corresponding to Media Type.



Advice

This is the approximate AWS upload bandwidth for a single PUT multiplied Number of Parts. (This metric is currently used for informational purposes only).

17.1.6. Storage class subsystem thresholds

These windows allow the user to manage storage class space thresholds which are specific to a storage subsystem. These windows are accessible only from the *Configured Storage Classes* window. The user must first select a storage class from the **Storage Classes List**. The selected storage class will be referred to as the "parent" storage class to the subsystem threshold window which is opened when the storage class is selected.

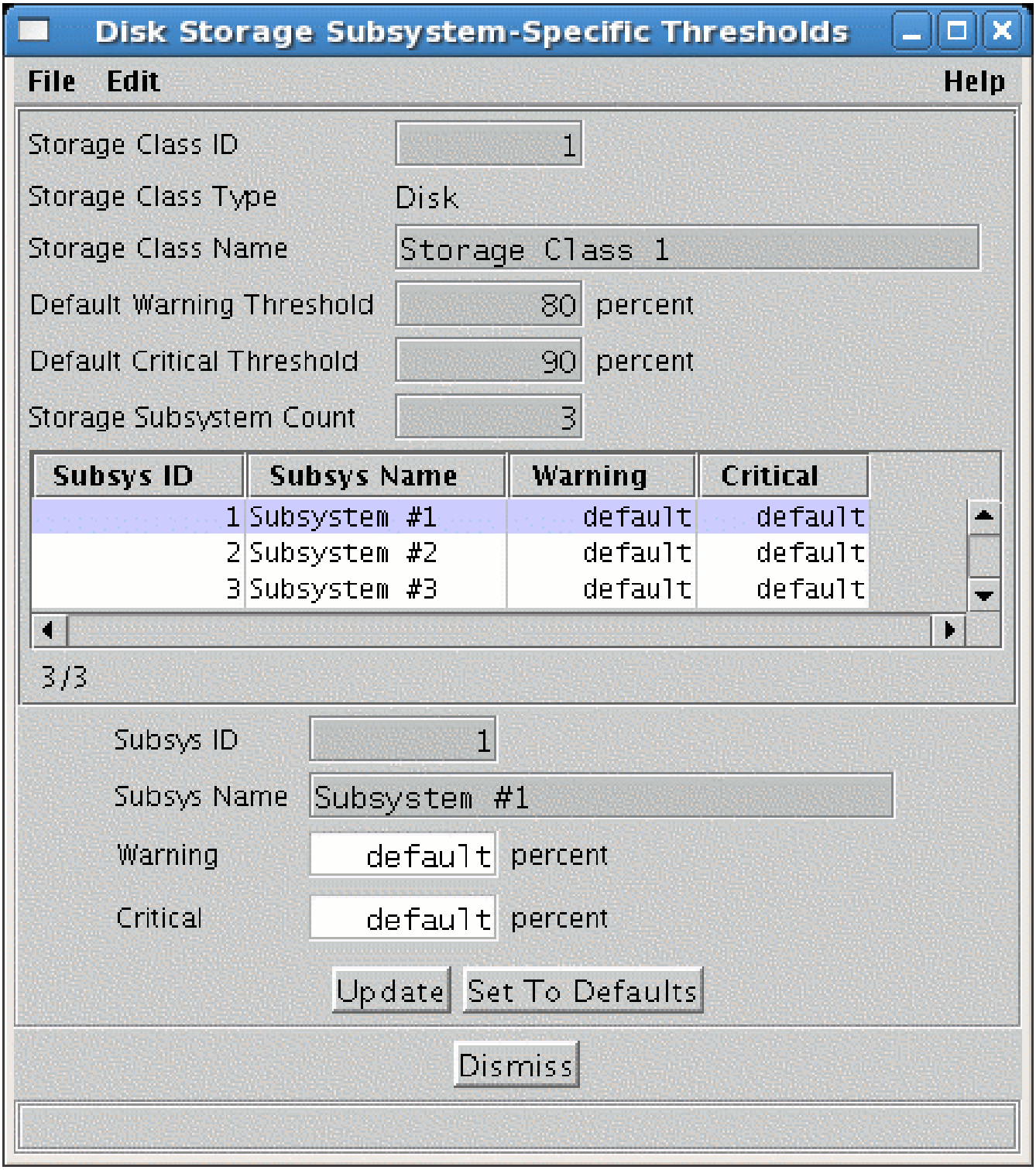
Each storage class configuration window has fields for **Warning Threshold** and **Critical Threshold**. By default, these thresholds apply to all storage subsystems using the storage class. However, the storage class thresholds can be overridden by configuring a subsystem-specific threshold allowing a storage subsystem to have its own customized set of **Warning Threshold** and **Critical Threshold** values.

After creating or modifying any thresholds, the changes will take effect after clicking the **Apply Config** button on the *HPSS Health and Status* screen.

Related information

Storage class threshold overrides

17.1.6.1. Disk Storage Subsystem-Specific Thresholds



This window is used to define warning and critical thresholds unique to a particular storage subsystem, overriding the values defined in the disk storage class. The user may modify either the **Warning** percent, **Critical** percent or both for one or more of the listed subsystems. Select a subsystem from the list and then modify the values on the lower portion of the window. When the new values have been entered, select **Update** to commit the changes. To remove the overridden threshold values, select the **Set To Defaults** button. The changes will be committed and displayed in the **Subsystem List** table.

After any modifications to the subsystem-specific thresholds, the changes will take effect after clicking the **Apply Config** button on the *HPSS Health and Status* screen.

Field descriptions

Storage Class ID

The ID of the storage class to which these subsystem-specific thresholds apply.

Storage Class Type

The type of storage class (disk or tape).

Storage Class Name

The descriptive name of the storage class.

Default Warning Threshold

The default warning threshold for this storage class expressed as a percentage of used space.

Default Critical Threshold

The default critical threshold for this storage class expressed as a percentage of used space.

Storage Subsystem Count

The number of storage subsystems in the HPSS system.

Subsystem List

A list of configured storage subsystems. Columns in the list include:

Subsys ID

The ID of the selected storage subsystem.

Subsys Name

The name of the selected storage subsystem.

Warning

The current warning threshold value for the selected subsystem. If the storage class defaults are to be used, the text `default` will be displayed.

Critical

The current critical threshold value for the selected subsystem. If the storage class defaults are to be used, the text `default` will be displayed.

Buttons

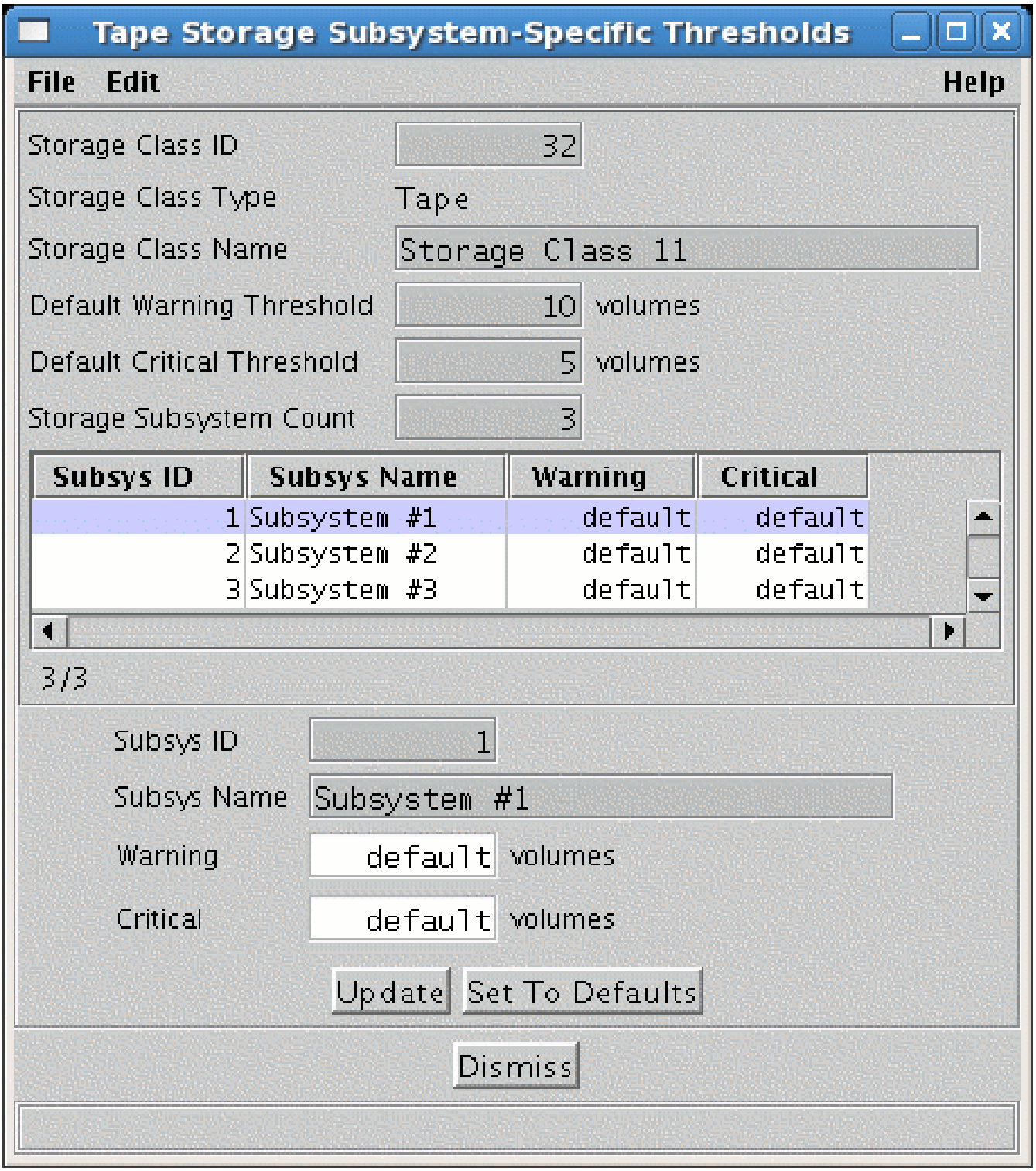
Set To Defaults

A button to remove the threshold override values from the selected storage subsystem. Select a storage subsystem row from the **Subsystem List**. The selected storage subsystem information will be displayed in the lower panel. Clicking the **Set To Defaults** button will commit the changes to the selected subsystem and `default` will be displayed for both the warning and critical thresholds.

Related Information

Storage class threshold overrides

17.1.6.2. Tape Storage Subsystem-Specific Thresholds



This window is used to define warning and critical thresholds unique to a particular storage subsystem, overriding the values defined in the tape storage class. The user may modify either the number of **Warning** volumes, **Critical** volumes or both for one or more of the listed subsystems. Select a subsystem from the list and then modify the values on the lower portion of the window. When the new values have been entered, select **Update** to commit the changes. To remove the customized threshold values, select the desired subsystem and click the **Set To Defaults** button. The changes will be committed and displayed in the **Subsystem List** table.

After any modifications to the thresholds, the changes will take effect after clicking the **Apply Config** button on the *HPSS Health and Status* screen.

Field descriptions

Storage Class ID

The ID of the storage class to which these subsystem-specific thresholds apply.

Storage Class Type

The type of storage class (disk or tape).

Storage Class Name

The descriptive name of the storage class.

Default Warning Threshold

The default warning threshold for this storage class expressed as a number of free volumes.

Default Critical Threshold

The default critical threshold for this storage class expressed as a number of free volumes.

Storage Subsystem Count

The number of storage subsystems in the HPSS system.

Subsystem List

A list of configured storage subsystems. Columns in the list include:

Subsys ID

The ID of the selected storage subsystem.

Subsys Name

The name of the selected storage subsystem.

Warning

The current warning threshold value for the selected subsystem. If the storage class defaults are to be used, the text `default` will be displayed.

Critical

The current critical threshold value for the selected subsystem. If the storage class defaults are to be used, the text `default` will be displayed.

Buttons

Set To Defaults

A button to remove the threshold override values from the selected storage subsystem. Select a storage subsystem row from the **Subsystem List**. The selected storage subsystem information will be displayed in the lower panel. Clicking the **Set To Defaults** button will commit the changes to the selected subsystem and `default` will be displayed for both the warning and critical thresholds.

Related information

Storage class threshold overrides

17.1.7. Changing a storage class definition



When changing a storage class's configuration, the Core and Migration/Purge Servers must be reinitialized or restarted in order for the changes to take effect. It is recommended to use the **Apply Config** button on the *HPSS Health and Status* window to reinitialize the servers.

If you're changing a storage class's definition as part of migrating to a new generation of storage media technology, refer to Storage technology replacement for details and a high level procedure.

A storage class definition can be changed by bringing up the *Storage Class Configuration* window and making the desired modifications. Most fields in an existing storage class configuration are directly editable. The few that aren't include **Data Stripe Length**, **VV Size**, **Stripe Transfer Rate**, and **Blocks Between Tape Marks**. When planning to alter various values in a storage class's configuration, take the following aspects into account:

Migration Policy

If files have been stored in a storage class without a migration policy, and a migration policy is subsequently configured for it, the files created before the addition of the policy will not be migrated. Use the mkmprec(7) utility to create migration records for these files so that they will migrate properly.

Migration or Purge Policies

When a migration or purge policy is modified, or when a storage class configuration is modified to use a different migration policy, the affected Migration/Purge Servers and Core Servers must be reinitialized or restarted in order for the new policy to take effect. It is recommended to use the **Apply Config** button on the *HPSS Health and Status* window to reinitialize the servers.

Optimum Access Size, Device I/O Rate, Average Latency

These values are used only in an advisory capacity by HPSS. A customer may write a special interface using the Client API that takes advantage of the COS Hints capability and, in so doing, uses these values. Generally, they can be changed at will, but should be set to accurately reflect their intended usage and the backend storage devices serving the storage class.

Maximum VVs To Write

Change this as desired.

Max Multiplier, Min Multiplier, Average Number of Storage Segments

These values can be changed as desired for *disk* storage classes. They do not apply to *tape* storage classes.

Media Type

Care must be taken when changing a storage class's **Media Type**.
Changing this value does not apply to existent HPSS virtual volumes (VVs); a VV's storage class configuration is set upon resource creation and remains immutable for the lifetime of the VV. A new **Media Type** will only apply to new VVs that are created after modifying the storage class.

After selecting a new **Media Type**, an icon () will appear next to various values in the *Storage Segment Size* and *Transfer Rate* sections that automatically update to match the newly selected **Media Type**. It's important to be attentive to the values that change. Closely vet them against the values that were in the storage class configuration prior to changing the **Media Type**. The values can change in unexpected and undesired ways. For example, an LTO-4 storage class's 32 MB **VV Block Size** may revert to 1 MB upon changing the storage class's **Media Type** to LTO-6. The new 1 MB **VV Block Size** is not likely to be the desired final value. See the Storage characteristics considerations section for guidance on configuring these storage class values.

Data Stripe Width and Parity Stripe Width

Care must be taken when changing a storage class's **Data Stripe Width** and **Parity Stripe Width**. File Hashing is not supported for striped storage classes, including RAIT storage classes. Therefore, if the storage class is modified to be striped (by changing **Data Stripe Width** to be greater than 1), then the storage class must not be used in any Class of Service which supports File Hashing. This means that the **Hash Algorithm** in any COS configuration which references a hierarchy which includes this storage class must be set to "None." Logical Block Protection (LBP) is not supported for RAIT. If the storage class is modified to support RAIT (by changing **Parity Stripe Width** to a non-zero value), then **Block Protection** must be turned off and the related fields in the **RAIT Options** section must be configured.

Changing this value does not apply to existing HPSS virtual volumes (VVs); a VV's storage class configuration is set upon resource creation and remains immutable for the lifetime of the VV. A new **Data Stripe Width** or **Parity Stripe Width** will apply only to new VVs that are created after modifying the storage class.

17.1.8. Deleting a storage class definition

Before deleting a storage class definition, be sure that all of the subsystem-specific warning and critical thresholds are set to `default`. If this is not done, one or more threshold records will remain in metadata and will become orphaned when the storage class definition is deleted.

To delete a storage class definition, ensure that no files exist in this storage class and it is no longer referenced in any hierarchy. All of the virtual volumes in this storage class should also be deleted, using the Delete Resources or Reclaim operation. Once the storage class is empty and it is not referenced from any hierarchy definition, the storage class definition can be removed.

See also the dump_acct_sum(7) utility, which can quickly verify the number of files in each COS.

It is recommended you call HPSS support for assistance with this operation.

17.2. Storage hierarchies

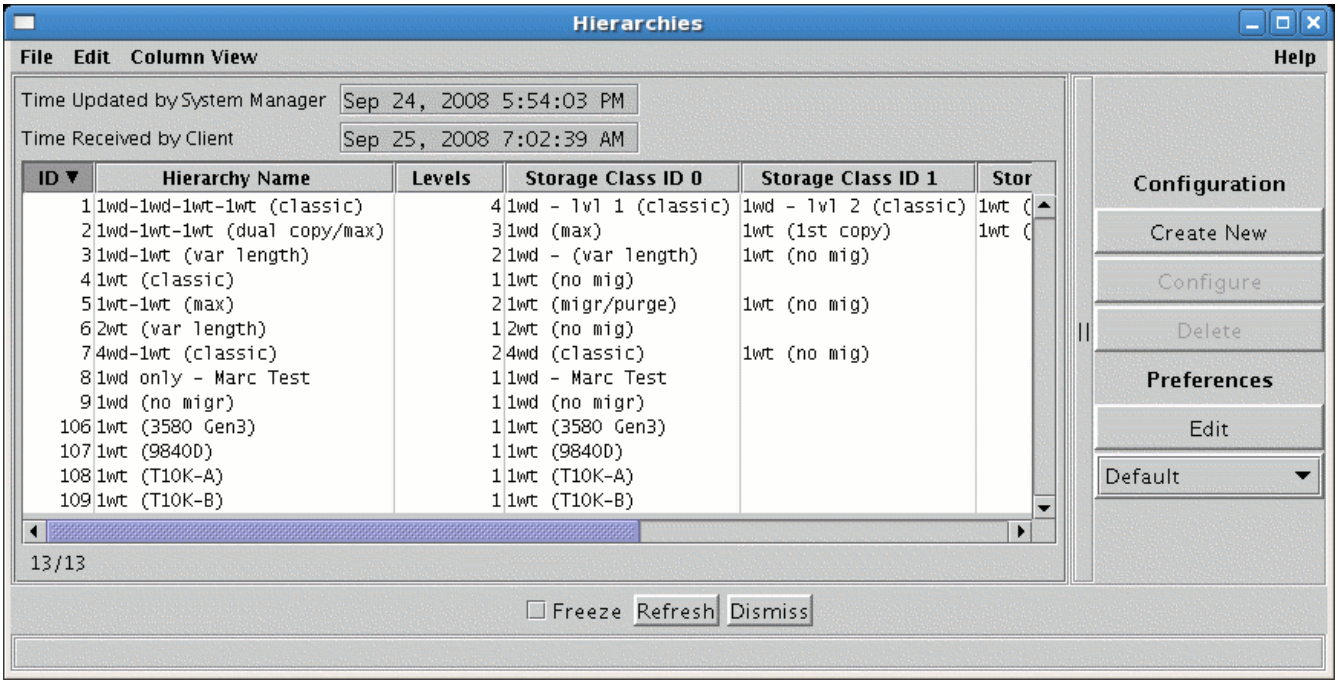
This section describes how to create, modify, and delete storage hierarchies.

17.2.1. Hierarchies window

An HPSS storage hierarchy can be created and managed using the *Hierarchies* window. This window is accessed from the *HPSS Health and Status* window's **Configure** menu, submenu **Storage Space**, item **Hierarchies.** Refer to HPSS Health and Status.

The following rules for creating storage hierarchies are enforced by the *Hierarchies* window:

- A storage class may be used only once per hierarchy.
- Disk migration may migrate to one or more target levels in the hierarchy. To create multiple copies of data, select multiple migration targets.
- Tape migration may only migrate to a single target level in the hierarchy.
- A given storage class may be a migration target for only one storage class above it in the hierarchy (two different storage classes in the hierarchy may not both migrate to the same storage class).



This window allows you to manage storage hierarchies. It lists all currently defined hierarchies and allows you to change existing hierarchies. New hierarchies can also be created from this window.



When a new hierarchy is created or updated, the Migration/Purge Servers and the Core Servers must be reinitialized or restarted in order for the addition to take effect. It is recommended to use the **Apply Config** button on the *HPSS Health and Status* window to reinitialize the servers.

Field descriptions

Hierarchy list

ID

The unique ID assigned to the storage hierarchy when it was configured.

Hierarchy Name

The descriptive name of the hierarchy.

Levels

The number of storage classes that have been configured into the hierarchy.

Storage Class ID 0

Storage Class ID 1

Storage Class ID 2

Storage Class ID 3

Storage Class ID 4

The name of the storage class at the given hierarchy level. HPSS arranges the storage classes into levels internally.

Configuration buttons

Create New

Open a *Storage Hierarchy* window with default values.

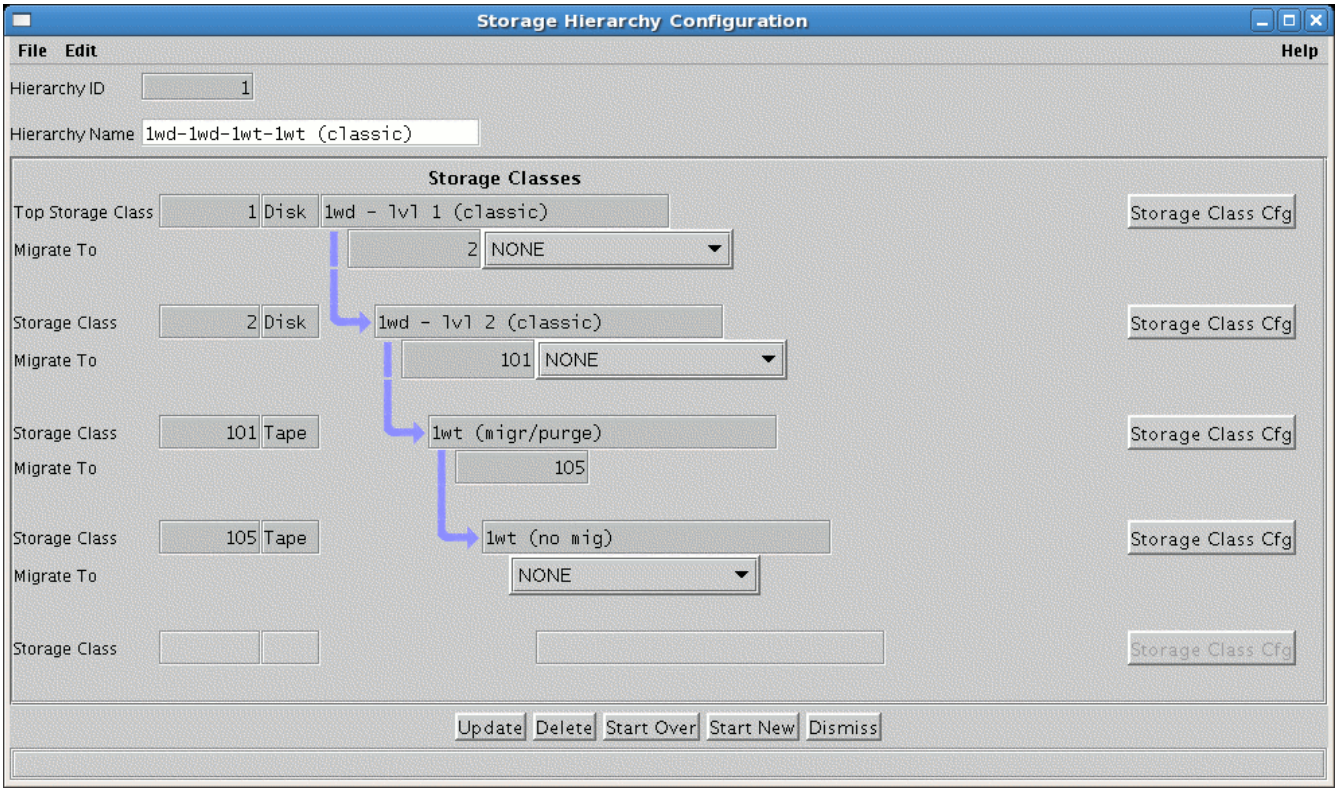
Configure

Open the selected storage hierarchy configuration for editing. One hierarchy from the list must be selected before this button is active.

Delete

Delete the one or more selected storage hierarchies.

17.2.2. Storage Hierarchy Configuration window



This window allows an administrator to manage a storage hierarchy.

A maximum of five storage classes can be configured into a hierarchy. A single storage class may appear in multiple hierarchies. *No hierarchy can reference the same storage class more than once.*

Note that in addition to defining the hierarchy, the appropriate migration policies must be

configured for each storage class in the hierarchy. Without the proper migration policies, neither migration nor duplicate tape copies will be maintained. See Migration policies.

Once a storage hierarchy has been created, it should not be deleted as long as there is a Class of Service entry associated with it. A Class of Service entry should never be deleted unless all files in that Class of Service have been deleted and no client is still attempting to use that Class of Service.

When a hierarchy is created or updated, the Core and Migration/Purge Servers must be reinitialized or restarted in order for the changes to take effect. It is recommended to use the **Apply Config** button on the *HPSS Health and Status* window to reinitialize the servers.

Field descriptions

Hierarchy ID

The ID associated with this hierarchy . Any unique, positive 32-bit integer value. The default value is the last configured ID plus 1.

Hierarchy Name

The descriptive name associated with this hierarchy. The default value is "Hierarchy <ID>".

Top Storage Class

The storage class at the highest level of the hierarchy.

After a storage class is selected, a new storage class drop-down list will appear in the **Migrate To** field. Selecting a storage class from the drop-down list will define the storage class to which the **Top Storage Class** will migrate, and an arrow will show the migration path from the **Top Storage Class** to the second storage class.

This process may be repeated to create more migration paths.

Depending on whether the **Top Storage Class** is disk or tape, either one or two new storage class drop-down lists will appear.



Advice

Currently, hierarchies with a top level tape storage class and a lower level object storage class are not supported.

Migrate To

The one or more storage class IDs to which the storage class will migrate. Initially, when creating a new hierarchy, these fields will not be visible. As the user configures the hierarchy, the **MigrateTo** values will be displayed. After a storage class has been selected from a drop-down list, the drop-down list will be replaced with a non-editable storage class ID corresponding to the user's selection. One or more new storage class drop-down lists will become visible.



Advice

Currently, object store storage levels can only be migration targets. No storage class can be configured to migrate from an object store storage class.

Storage Class

The ID number of the storage class at the second through the fifth level of the hierarchy. See description above for **Top Storage Class**.

Storage Class Cfg button

Clicking this button will open the *Storage Class Configuration* window corresponding to the storage class name to the left of the button.

17.2.3. Changing a storage hierarchy definition

A storage hierarchy definition can be changed by bringing up the *Storage Hierarchy Configuration* window and making the desired modifications. Note that the only modification that is possible to an existing storage hierarchy is the inclusion of additional storage levels at the bottom of the hierarchy. All other modifications require that the hierarchy be deleted and recreated.

When adding a migration target to the bottom of a storage hierarchy, be sure that the migrating storage class is configured with appropriate migration and purge policies. Without migration and purge policies, no migration and purge will take place even if target levels exist in the hierarchy. Furthermore, any pre-existing data in the storage class (data which was written into the storage class before the storage class was configured to use migration and purge policies) will not be migrated and purged even after the migration and purge policies are added. The migration and purge policies are honored only for data written after the policies are put into effect. To migrate and purge data written before these policies became effective, use the mkmprec(7) utility.

17.2.4. Deleting a storage hierarchy definition

The rules that apply for deleting a Class of Service also apply for deleting a storage hierarchy because in most systems a one-to-one relationship exists between hierarchies and Classes of Service. Before deleting a hierarchy, all the files in all of the Classes of Service that use the hierarchy must be deleted. Failure to do this will render the files unreadable.

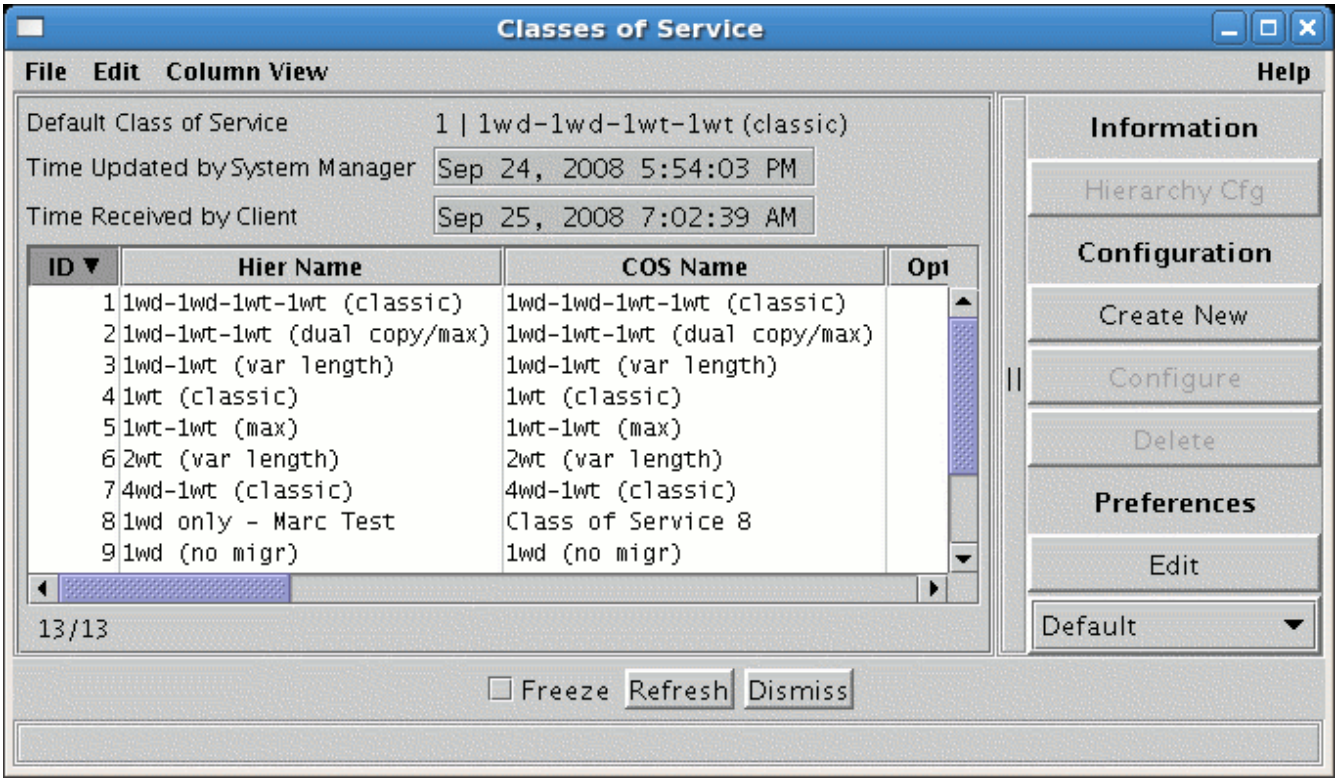
It is recommended you call HPSS support to assist with this operation.

17.3. Classes of Service

This section describes how to configure Classes of Service.

17.3.1. Classes of Service window

A COS can be created and managed using the *Classes of Service* window.



This window lists the Classes of Service that are currently configured. It also allows an administrator to update and delete existing Classes of Service and to add a new Class of Service. This window is displayed by selecting the **Configure** menu from the *HPSS Health and Status* window, then selecting the **Storage Space** submenu, and then selecting the **Classes of Service** submenu item.



When a new Class of Service is created or updated, the Migration/Purge Servers and the Core Servers must be reinitialized or restarted in order for the addition to take effect. It is recommended to use the **Apply Config** button on the *HPSS Health and Status* window to reinitialize the servers.

Field description

Default Class of Service

The default Class of Service. This is displayed for information only and may not be changed from this window. It may be changed from the *Global Configuration* window.

Class of Service List

The central panel in this window shows the list of configured Classes of Service. The columns display the fields from the *Class of Service Configuration* window discussed in the following section.

Information buttons

Hierarchy Cfg

Open a *Storage Hierarchy Configuration* window for the hierarchy associated with the selected Class of Service.

Configuration buttons

Create New

Open a *Class of Service* window containing default values for a new Class of Service.

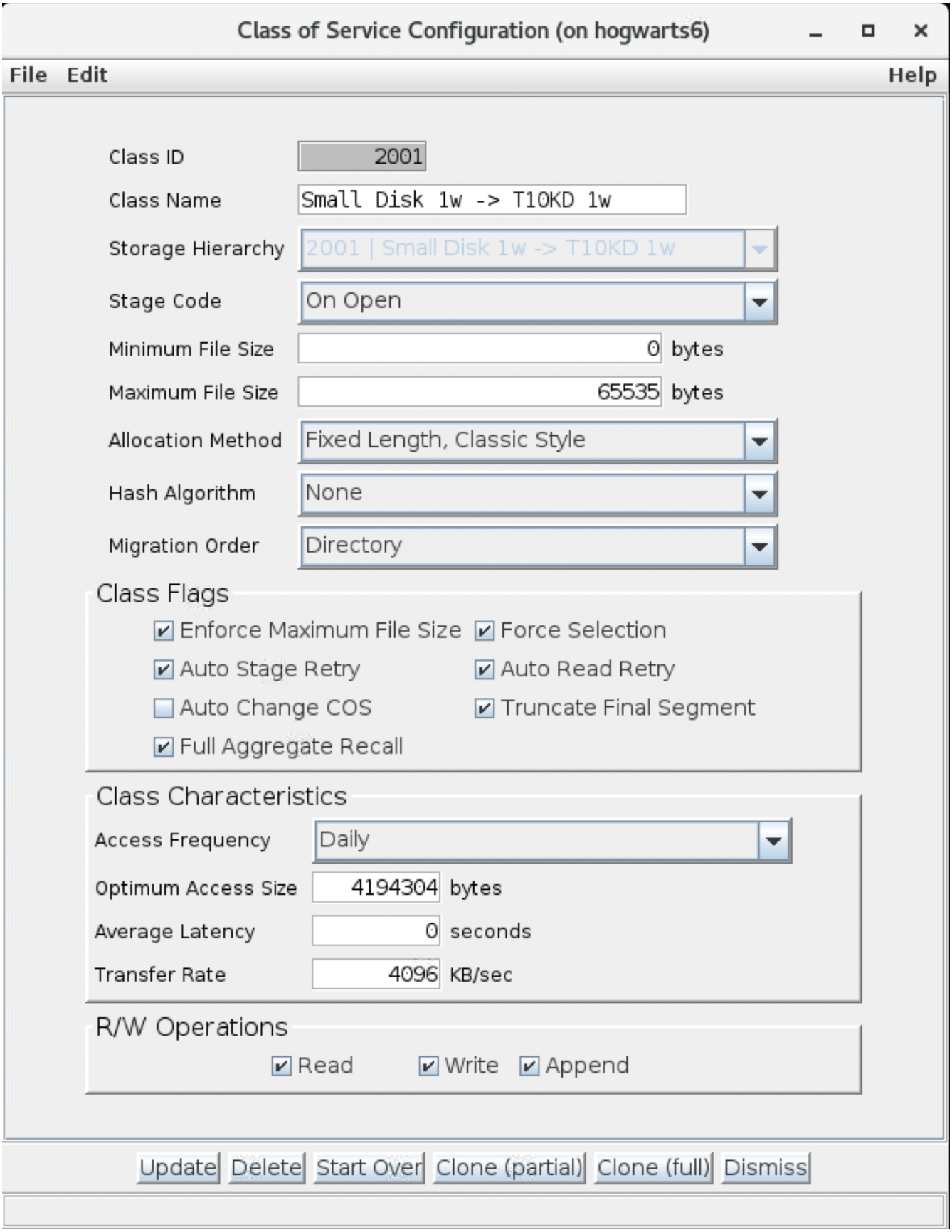
Configure

Open the selected Classes of Service configurations for editing.

Delete

Delete the selected Classes of Service.

17.3.2. Class of Service Configuration window



This window allows an administrator to manage a Class of Service.

Field descriptions

Class ID

The unique integer ID for the COS. Any positive 32-bit integer value.

Class Name

The descriptive name of the COS. A character string up to 31 bytes in length. The default value is "Class of Service <ID>".

Storage Hierarchy

The name of the storage hierarchy associated with this COS.

Stage Code

A code that indicates the file staging options. Valid values are `On Open`, `On Open Async`, `No Stage`, and `On Open Background`. For the first COS created in a new HPSS system, the default value is `On Open`. For all subsequently created COSs, the default value is the same as the most recent COS configured.



Advice

Changing the Stage Code should be done with care. See Changing a Class of Service definition for detailed information on each of the choices for Stage Code.

Minimum File Size

The size, in bytes, of the smallest bitfile supported by this COS. Valid values are any positive 64-bit integer value.

Maximum File Size

The size, in bytes, of the largest bitfile supported by this COS. Valid values are any positive 64-bit integer value.

Allocation Method

How disk storage segments will be allocated in the storage class at the top level of the hierarchy used by this COS. Ignored if the top level storage class is tape. Three allocation methods are available:

`Fixed Length, Classic Style`

All storage segments allocated for a single file, except possibly the last segment of the file, are the same size. This size is computed at file creation time to be the size which wastes the least disk space, based on the file size, the minimum and maximum segment sizes configured for the storage class, and the average number of segments per file configured for the storage class. In order to take maximum advantage of this allocation method, the application creating the file must specify the file size in its file creation hints.

The application can force this fixed segment size to be the maximum segment size configured for the storage class by specifying HINTS_FORCE_MAX_SSEG in the file creation hints.

If the application does not force the segment size to be the max size and it does not provide hints telling the Core Server the file size, then the Core Server will use the minimum segment size configured for the storage class.

If the file is purged completely from the top level and later restaged to the top level, a new segment size will be computed at stage time based on the current file size.

`Fixed Length, Max Style`

The size of each storage segment allocated for a file, except possibly the last segment, is the maximum segment size configured for the storage class.

`Variable Length`

The size of the first storage segment allocated for a file is the minimum segment size configured for the storage class. The size of each successive storage segment is twice that of the preceding segment, up to the maximum segment size configured for the storage class. Once the max segment size is reached, each remaining segment except possibly the last is also the max segment size.

Under all three allocation methods, the last segment of the file may be truncated to a size to best fit the actual data contained in the segment. See the **Truncate Final Segment** option in the **Class Flags** section below.

Hash Algorithm

This is the default hash algorithm to be used when generating digests for files in this COS as they are migrated or copied to tape, or written directly to tape. The following hash algorithms are supported.

- SHA1
- SHA224
- SHA256
- SHA384
- SHA512
- MD5
- CRC32
- ADLER32

This functionality is supported on storage hierarchies that contain a disk storage class at the initial level, with that storage class migrating to one or more tape storage classes. Because of the sequential nature of the supported hash algorithms, digest generation is currently limited to data being migrated or copied to non-striped tape volumes.

Another limitation involves the ability of HPSS to have the most current data spread over multiple levels in the storage hierarchy. This most often occurs when a file is appended after a duration long enough for the previous data to be migrated and purged from the initial disk storage level. If this occurs the new data will eventually be migrated, but the original data would already be valid at a lower level and will not be migrated. In this case, the file digest will not be generated or verified and a warning message will be issued.

This functionality is also supported when writing directly to tape. The Class of Service must contain a single stripe tape as the top tier storage class, and the file must be written from beginning to end without any backseeks or parallel write operations. Concurrent access to the file will invalidate the hashing process. Subsequent writes to the file after after the initial checksum has been generated will invalidate the file hash. The file hash will be finalized when

the file is closed and compared against any user hash or stored as a generated checksum.

A user-provided digest can be associated with a file and a system generated digest can be obtained using programming interfaces.

Migration Order

The migration ordering criteria for this COS. The following orderings are supported:

Create Time (Default)

Migration will be ordered by file create time. This is the default.

Directory

Migration will be ordered by directory. Each directory's files eligible for migration will be grouped together. The depth of the directory grouping is one deep. Thus, files in a sub-directory will be grouped with other files in the sub-directory and not with files in the parent directory.

The objective is to co-locate data on tape, with either the directory of the files or the files' create times being the co-location criteria. A site may want to combine this setting with *COS Class Flag* **Full Aggregate Recall** to potentially minimize the number of tape operations required to bring a directory's files back from tape. Of course, the migration policy will continue to honor the migration policy settings for when a file is eligible for migration and how often migration will run. Refer to the Disk Migration Policy configuration section for further disk migration options.

When modifying the **Migration Order**, the System Manager will notify the Migration/Purge Server about the change. The change won't affect in-flight migrations, but subsequent migrations will use the new **Migration Order** setting.

Class Flags

Enforce Maximum File Size

When set, a bitfile larger than the **Maximum File Size** cannot be created in this COS.

Force Selection

A flag to determine how a COS will be selected. If ON, a client must explicitly select this COS in order to have a file assigned to it; if the client merely supplies general COS hints for a file, this COS will not be selected.

Auto Stage Retry

When this flag is turned on, and a valid secondary copy of the data exists, and a stage from the primary copy fails, HPSS will automatically retry the stage using the secondary copy.



Advice

*To stage from an object store storage class the **Stage Allowed** flag must also be enabled in the storage class configuration. See Object Store Storage Class Configuration for additional information.*

Auto Read Retry

When this flag is turned on, and a valid secondary copy of the data exists, and an attempt to

read the first copy fails, HPSS will automatically retry the read using the secondary copy. This retry will not be attempted if any data was transferred between the client and HPSS during the failure.



*For **Auto Stage Retry** and **Auto Read Retry** to work properly, the COS must contain at least two copies of the files and at least one valid second copy must have been created during HPSS migration processing. If any tape storage class in the COS has a migration policy, that policy must be configured for **Migrate Files** or **Migrate Files and Purge.***

Auto Change COS

When this flag is turned on, upon disk migration, files within this Class of Service (COS) that fall outside the **Minimum File Size** and **Maximum File Size** will have their COS changed to a target COS that has the appropriate file size range. The Migration/Purge Server initiates the change in Class of Service and the Core Server performs it as a background task. COSes with the **Force Selection** flag on will not be considered for the target COS. If multiple COSes meet the criteria, then the Migration/Purge Server will select the first COS that meets the criteria; if no COSes meet the criteria, then the file will remain in the current COS and be migrated according to the current COS's hierarchy. If the Core Server's **COS Copy to disk** flag, described in Core Server-specific configuration, is not on and there is a tape level in the target COS's hierarchy, then the Core Server will not automatically copy the file to the disk level and will instead copy to the tape level. To take advantage of migration policies for the new COS's hierarchy (like tape aggregation), it is recommended that you turn **COS Copy to disk** on.

Truncate Final Segment

A flag to influence whether the final segment of files will be truncated at file close time in order to save disk space.

When a file is closed, its final segment may be truncated to the smallest segment size which is valid for the storage class and will still hold the actual data written to the segment. This size will be a power of two multiple of the minimum segment size configured for the storage class. The segment will later be expanded to its original size if the file is reopened and new data is written to the segment or to a point in the file beyond the segment. Truncation of the final segment is performed only on the file data at the top level of the storage hierarchy and only if the storage class at the top level is disk.

Truncation of the final segment is controlled by the NOTRUNC_FINAL_SEG flag in the bitfile descriptor; if this flag is off, truncation is performed at file close, and if it is on, truncation is not performed.

The COS **Truncate Final Segment** flag influences whether the NOTRUNC_FINAL_SEG flag in the bitfile descriptor is set at file creation time. The user may specify a creation hint to turn truncation off even if the COS allows truncation, but truncation can not be turned on if the COS prohibits it. If the COS **Truncate Final Segment** flag is on, then at file creation time the NOTRUNC_FINAL_SEG flag will be set to off, unless the user specifies the NOTRUNC_FINAL_SEG hint. If the COS **Truncate Final Segment** flag is off, then the NOTRUNC_FINAL_SEG flag will be set to on at file creation time, and the user may not override this.

Users may also turn on the NOTRUNC_FINAL_SEG flag in their own files after file creation by use of the hpss_SetFileNotrunc Client API function, but they may not turn off the flag. Authorized callers may use the hpss_SetFileNotrunc function to turn the flag on or off for any file. Authorized callers are those such as hpssssm who have CONTROL permission on the Core Server's Client Interface ACL.



Advice

Truncation of the final segment normally saves disk space. However, if a file is frequently written, closed, reopened, and appended to, the repeated truncation and re-expansion of the final segment could result in fragmentation of the disk, as the space for the expansion is not guaranteed to be adjacent to the original segment. In addition, it is possible that the only extent available for the expansion is larger than needed, so some space could actually be wasted.

*If it is known that the users of a COS do frequent appends to their files, it is better to turn the COS **Truncate Final Segment** flag off to avoid these potential fragmentation and space waste issues. If no users do frequent appends, or if those who do can be relied upon to turn truncation off for their own files, or if the system administrator can easily identify files which are frequently appended and can turn off truncation on them individually, then the site might want to take advantage of the space savings for the remaining files and leave **Truncate Final Segment** on in the COS definition.*

One additional consideration is that truncating the final segment incurs a small performance penalty. In development testing, this penalty was measured at 14% for writing 100 files with truncation in comparison to writing them without truncation; this value may vary under different site configurations.

Full Aggregate Recall

When this flag is turned on, and a stage is requested for a file which resides within a tape aggregate, then all of the eligible files within the aggregate are staged. Ineligible files are those which have been deleted and those which already have data staged at a higher level.

Class characteristics

Access Frequency

The frequency, on average, for accessing files in this COS. Valid values are `Hourly`, `Daily`, `Weekly`, `Monthly`, or `Archive`.

Optimum Access Size

The suggested number of bytes that should be written at one time for maximum efficiency. (Not currently used by HPSS; may be used by site-specific applications.)

Average Latency

The average time, in seconds, that elapses between the time a transfer request is accepted for processing and the time the data transfer begins.

Transfer Rate

The average throughput (in KB per second) that can be transferred using this COS.

R/W operations

Read

If ON, files in this COS are readable, subject to the permissions on each individual file.

Write

If ON, files in this COS are writable, subject to the permissions on each individual file.

Append

If ON, files in this COS can have new data appended to them, subject to the permissions on each individual file. This check box may only be ON if the **Write** check box is also ON. If the **Write** check box is OFF, the **Append** check box is disabled.

In all cases, whether an individual file can be read or written will be determined by the permissions on that particular file. It may be the case that a COS is RW, but certain files within that COS may have their permissions set such that they can be neither read nor written.

17.3.3. Changing a Class of Service definition

A COS definition can be changed by bringing up the *Class of Service Configuration* window and making the desired modifications. Fields on the *Class of Service Configuration* window that can be changed without major impact are the **Access Frequency**, **Optimum Access Size**, **Average Latency**, and **Transfer Rate**. These fields are advisory in nature and are currently used only by the Client API. Users of custom HPSS interfaces will have to assess the impact of any change.

The **Maximum File Size** can be changed, but care should be exercised when doing so. Increasing this value may result in storing files that are inappropriate for the hierarchy supporting this COS. For example, suppose a hierarchy is defined with disk at the top level with a storage segment size of 64 KB and the **Maximum File Size** in this COS is set to 1 MB. Changing the maximum file size to 2 GB in this class would be inappropriate because 32,768 storage segments (each 64 KB in size) would be needed to contain such a file.

Associated with the **Maximum File Size** is the **Enforce Maximum File Size** flag. Changing this can have a significant impact. Turning the flag on constrains files that are already larger than the **Maximum File Size** to their current size. Existing smaller files will be constrained to the **Maximum File Size**.

Changing **Minimum File Size** can have an impact on COS selection. Currently, the PFTP and FTP interfaces use the **Minimum File Size** to select an appropriate COS based on file size.

Changing the **Stage Code** should be done with care:

- The **On Open** option stages files to the top of the hierarchy when the file is opened, and is synchronous. The open operation returns when the stage operation completes.
- The **No Stage** option can have a significant impact on system performance. Repeated reads of files that have been migrated will generally be satisfied from lower levels in the hierarchy where files are likely to be on tape. If users are writing only parts of a file at a time with significant delays between the writes, migration may result in the file being stored in a more fragmented manner on tape.

- The **On Open Async** option causes stage operations to be started when files are opened, but open operations do not wait for stages to complete; they return to their callers immediately. If a caller then reads one of these files, the read operation will block until the stage is complete.
- The **On Open Background** option causes an open operation to complete immediately, returning a special error code that indicates that the stage is not complete. The caller can then poll HPSS for completion of the stage operation before reading the file. Normally this polling is done automatically by the Client API, but Client API polling can be turned off which will allow the client to do the polling.

Changing the **Auto Stage Retry** flag has no adverse side effects. It is recommended that this flag be enabled for all COSs using multiple copies.

The hierarchy associated with a COS cannot be changed without deleting all files in this COS or moving them to another COS. Failure to observe this condition will likely result in lost and corrupted files. Any changes to an existing COS other than changing the hierarchy ID can be put into effect by clicking the **Apply Config** button on the *HPSS Health and Status* window.

When modifying the **Migration Order**, the System Manager will notify the Migration/Purge Server about the change. The change won't affect in-flight migrations, but subsequent migrations will use the new **Migration Order** setting.

17.3.4. Deleting a Class of Service definition

To delete a COS, you must ensure that all files in the COS have been either deleted or moved to another COS and that all configuration references to the COS have been removed. These configurations include the Global Configuration's Default Class of Service, Storage Subsystem's Default COS Override, and every fileset's associated Class of Service.

Deleting a Class of Service is an unusual administrative action and HPSS does not provide utility programs to delete all the files in a given COS or move all files in a given COS to another COS. A certain amount of site-specific ad hoc planning and usage of available tools will be necessary to identify, remove, or move all files from a Class of Service. The dump_acct_sum(7) utility program provides a count of the number of files in all Classes of Service based on the accounting metadata and was developed primarily so there would be a reasonable way to know if a Class of Service was empty.

To delete the COS definition, use the **Delete** button on the appropriate *Class of Service Configuration* window.

17.3.5. Changing a file's Class of Service

The Core Server provides a means to change the Class of Service of a file. The Core Server moves the body of the file as appropriate to media in the destination Class of Service, then allows the usual migration and purge algorithms for the new Class of Service to apply. The file body is removed from the media in the old Class of Service. If large numbers of files on tapes are changed, an opportunity to retire, repack, and reclaim tapes in the old Class of Service may arise.

The hpsssetcos(7) utility can be used to initiate the COS change request of a file as follows:

```
# hpsssetcos <COSId> <fullPathName>
```

COS changes are performed using standard HPSS internal copy mechanisms. These do not verify the file checksums during the copy. Sites should record the checksums of files to be COS-changed and verify them with hpsssum after the COS change is complete.

17.3.6. Canceling a Class of Service change request

A Class of Service change request can be canceled in order to deal with a persistent problem in changing the file.

The hpsssetcos(7) utility can be used to cancel the COS change request as follows:

```
# hpsssetcos 0 <fullPathName>
```

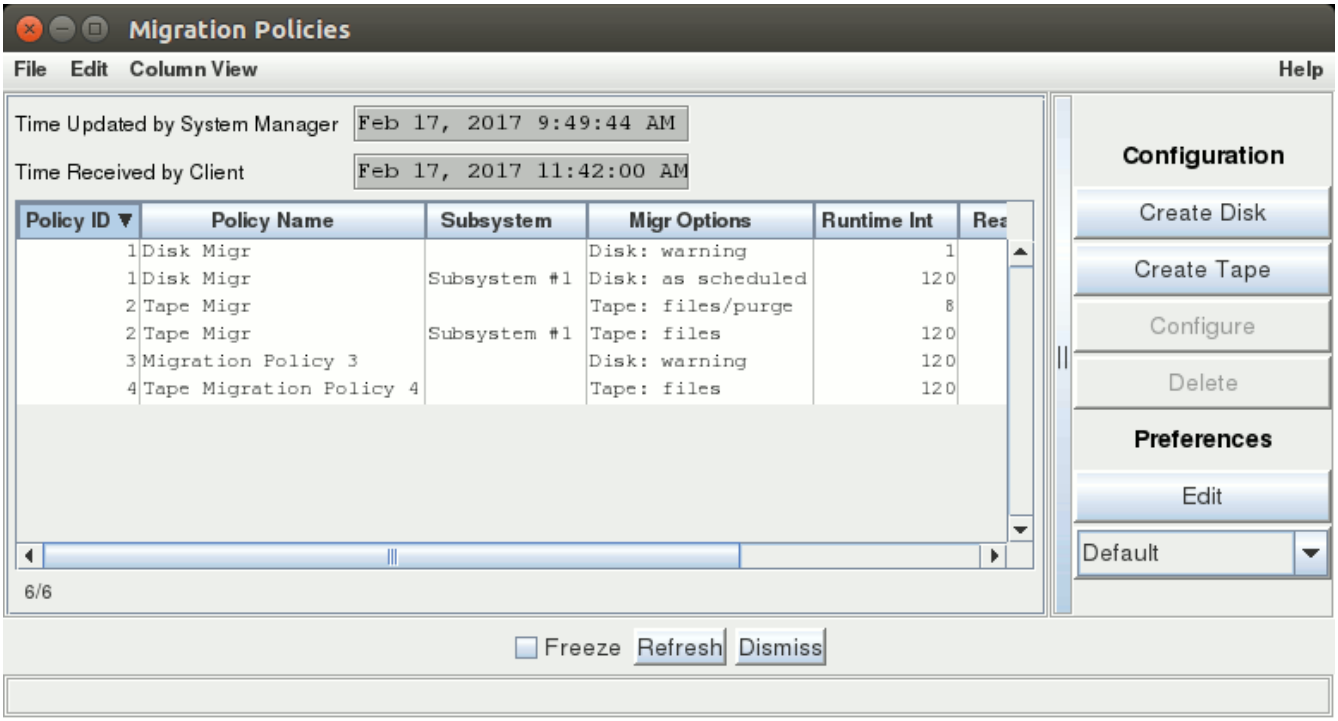
The associated Core Server COS change record will be removed immediately. However, this request may fail if the existing COS change request is currently being processed by one of the Core Server's COS change threads. In this case, the user should attempt to reissue the request later.

17.4. Migration policies

A migration policy is associated with a storage class and defines the criteria by which data is migrated from that storage class to storage classes at lower levels in the storage hierarchies. Note, however, that it is the storage hierarchy definitions, not the migration policy, which determine the number and location of the migration targets. Also, note that a storage class may be used in multiple hierarchies and may have different migration targets in each.

A basic migration policy must be assigned to any disk or tape storage class that requires migration services. This basic policy determines the migration parameters for the storage class across all storage subsystems. Storage subsystem-specific migration policies may be added which override the basic policy in the selected subsystems. The storage subsystem-specific migration policies share the migration Policy ID and Policy Name with the basic policy.

17.4.1. Migration Policies window



This window displays a list of all the configured migration policies.

Both basic and subsystem-specific migration policies are created and managed using the *Migration Policies* window. The basic policy must be created before creating any subsystem-specific policies.

The fields in the basic policy are displayed with default values. Change any fields to desired values as needed. Click on the **Add** button to write the new basic policy to the HPSS metadata.

To configure a subsystem-specific policy, select an existing basic policy and click on **Configure**. The basic policy will be displayed in a tabbed window with a tab for each storage subsystem for which subsystem-specific thresholds have been created. Click on the tab for the desired storage subsystem and make the necessary subsystem-specific changes, and then click on the **Update** button. To add a specific threshold configuration for a new subsystem, click the **New Subsystem Policy** button.



*When a migration policy is added to an existing storage class, the Migration/Purge Servers for that storage class and the Core Servers must be reinitialized or restarted in order for the policy to take effect. It is recommended to use the **Apply Config** button on the HPSS Health and Status window to reinitialize the servers.*

When HPSS needs to apply a migration policy to a subsystem, it first searches for a matching subsystem policy. If none is found, HPSS uses the basic policy. You can consider the basic policy to be the default for all subsystems which do not have policies of their own.

If a policy has one or more subsystem-specific overrides, the basic and subsystem-specific policies will be listed separately. Each entry will have the same ID, but the **Subsystem** field will be blank for the basic policy and contain the subsystem ID for each subsystem-specific override.

After changing any migration policy information, the changes will take effect after clicking the **Apply Config** button on the *HPSS Health and Status* screen.

Field descriptions

Policy List

Policy ID

The unique ID number assigned to this migration policy.

Policy Name

The name assigned to this migration policy.

Subsystem

If the policy is a subsystem-specific override, this shows the subsystem to which it applies. This field will be blank for basic policies.

Other Migration Policy List columns

The remaining columns provide the same information that can be found in Disk Migration Policy configuration and Tape Migration Policy configuration windows.

Configuration buttons

Create Disk

Opens a *Disk Migration Policy* window with default values.

Create Tape

Opens a *Tape Migration Policy* window with default values.

Configure

Opens the selected migration policy for editing.

Delete

Deletes the one or more selected migration policies.

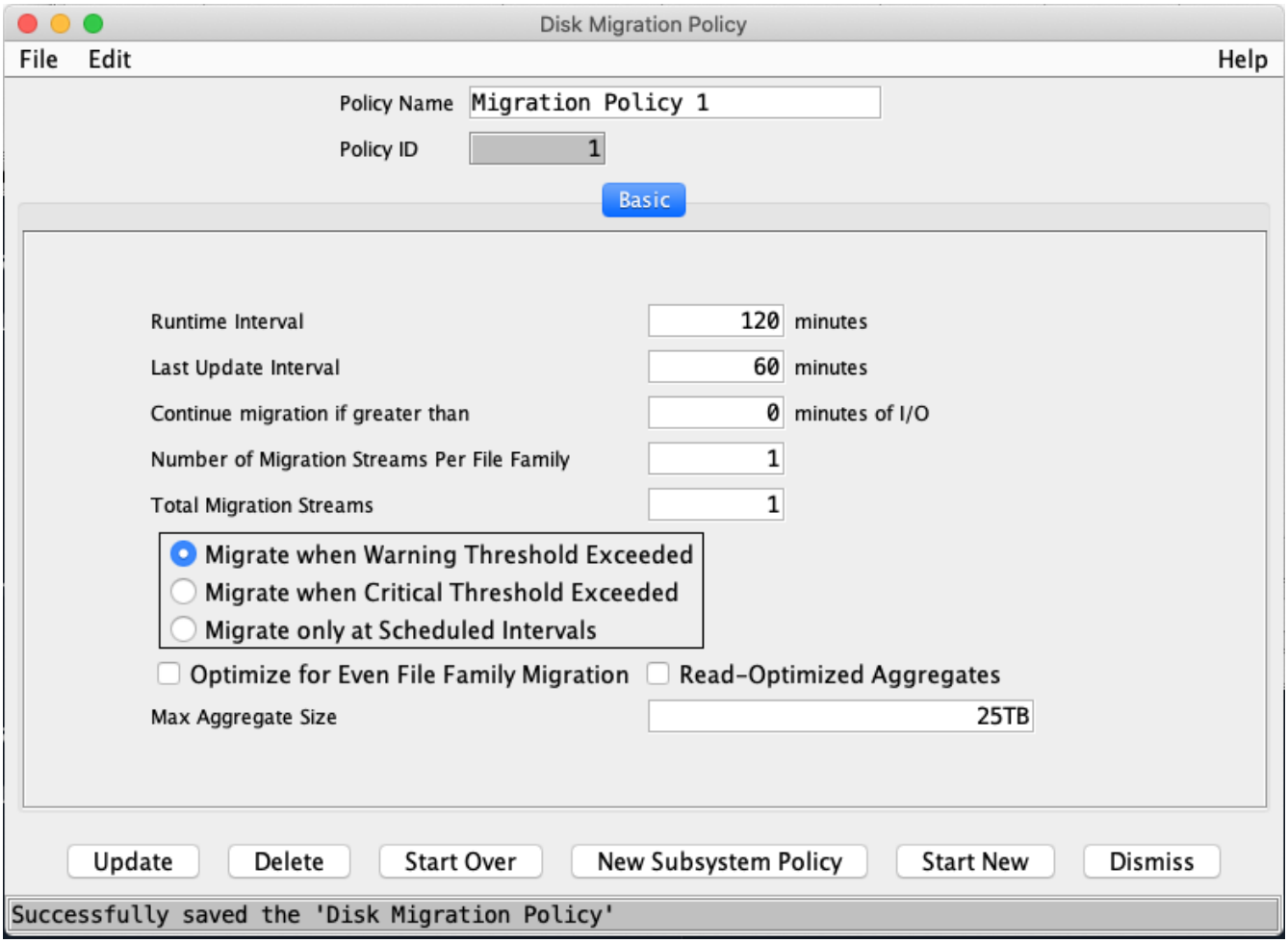
Related information

Section Migration/Purge Server

17.4.2. Migration policy configuration

This section describes the configuration, update, and deletion of migration policies.

17.4.2.1. Disk Migration Policy configuration



This window allows an administrator to manage disk migration policies and their subsystem-specific overrides.

Subsystem-specific policies define migration rules to be applied on a subsystem basis instead of using the default migration policy. When a migration run occurs and the storage class's migration policy does not have a subsystem-specific migration policy with the same subsystem ID, then the basic migration policy will be applied; otherwise, the matching subsystem-specific migration policy will be used. Click the **Create Disk** button on the *Migration Policies* window to create a new policy. To bring up the configuration for an existing policy, select the policy from the list and click on the **Configure** button.

Field descriptions

Policy Name

The descriptive name of the migration policy.

Policy ID

A unique ID associated with the migration policy.

Runtime Interval

The number of minutes to wait between the end of one migration run and the start of the next. This can be in the range [0 to 4,294,967,295].



Advice

*The **Runtime Interval** is the maximum number of minutes from the end of one migration run to the start of the next. Migration needs to run often enough so the storage class to which the policy is assigned does not fill up. Migrations can also be started manually from the Active HPSS Storage Classes window.*

Last Update Interval

The number of minutes that must pass since a file was last updated before it can become a candidate for migration.

Continue migration if greater than <#> minutes of I/O

This setting allows disk-to-tape migration to continue if it will result in I/O that runs for more than the specified amount of time; otherwise, if there's not enough data to meet the threshold, then the newly eligible files will be migrated on the next **Runtime Interval**, thus freeing up drives. For some customer workloads, this might be useful for reducing dismount/remount activity during migration. The overall idea is to avoid further tape mount activity unless enough data has accumulated to warrant the cost of mounting the tapes.

Consider, for example, an HPSS instance with multiple file families and a slow ingest rate for each family. The slow, consistent influx of files in each family causes migration to continue running, migrating just a few files at a time per family. This might be acceptable for customers that have drives dedicated to migration or are primarily concerned with getting files migrated as soon as possible. However, other customers might prefer for migration to start, migrate all the currently eligible files, and then only continue migrating data if some minimum amount of data has accumulated.

A value of "0" will be the default and will represent the default migration behavior: that disk-to-tape migration will continue as long as existent files are eligible for migration. A nonzero value will represent the minimum time for which migration I/O will need to run in order to continue migration.

This field will be ignored for disk-to-disk only hierarchies (that is, no tape).

To calculate the minutes of I/O, the MPS will look at the storage class transfer rate of the tape storage classes in each level of the hierarchy involved in the migration; the MPS will pick the slowest transfer rate to use for the calculation.

For example, in a disk to dual copy tape hierarchy, the transfer rate used to calculate whether to continue will be the slower of the two tape storage class transfer rates. If the continuation threshold is 1 hour and the primary copy storage class has a transfer rate of 300 MB/s and the secondary copy storage class has a transfer rate of 100 MB/s, the migration will continue if there is at least 360 GB to migrate (1 hour migrating at 100 MB/s).



Advice

Each file family migration will look for this field and determine whether or not it should continue. Thus if one family has enough data to continue, that family will continue to migrate; but if another family doesn't have enough, migration of that family's data will be delayed until the next runtime interval.

Verify that the Transfer Rate is correct in your storage classes.

Number of Migration Streams Per File Family

The number of migration streams to be allocated to each file family per copy level. This value effectively determines how many file families can be migrated simultaneously for a given copy level. For example: **Number of Migration Streams Per File Family** is set to "2" and **Total Migration Streams** is set to "10"; up to five families will be migrated simultaneously (five families with two streams working on each family yields a total of ten migration streams).

Since each migration stream can potentially mount tapes equal to the stripe width of the storage class, the total number of tape mounts for a file family during migration can be calculated as: **Number of Migration Streams Per File Family** × **Stripe Width**. For example, if a file family uses a storage class with a stripe width of 3 and has 2 migration streams allocated, up to 6 tapes could be mounted simultaneously for that file family's migration at a given copy level.

Table 26. Practical Examples: Migration Streams and Tape Mounts

Configuration	Stripe Width	Streams Per Family	Total Streams	Copy Levels	Max Tape Mounts	Notes
Single Copy, Non-Striped	1	2	10	1	10	Basic configuration: 10 streams × 1 tape each = 10 mounts
Single Copy, Striped	4	2	10	1	40	Each stream mounts 4 tapes: 10 streams × 4 tapes = 40 mounts
Dual Copy, Non-Striped	1	2	10	2	20	10 mounts per copy level: 10 + 10 = 20 total mounts
Dual Copy, Striped Primary	4	2	10	2	50	Primary: 10 × 4 = 40, Secondary: 10 × 1 = 10, Total = 50 mounts
Dual Copy, Both Striped	4	2	10	2	80	Both levels striped: (10 × 4) + (10 × 4) = 80 mounts

The table above shows how different configuration choices afect the total number of tape mounts during a migration for a single hierarchy. Some observations:

- A strip width of 4 means each migration stream requires 4 tape drives instead of 1
- Each copy level in the hierarchy runs a migration independently, so tape mounts accumulate accross all levels
- Primary and secondary copies can have different stripe widths (e.g., 4-wide primary, 1-wide secondary)

Multiple Hierarchies and Parallel Migration

The mount calculations shown in the table apply to a single hierarchy. However, as noted earlier in this section, "Each hierarchy within a disk storage class is migrated in a seperate but parallel fashion." This means that if a disk storage has multiple hierarchies configured, and there are files eligible for migration in each hierarchy, the tape mount requirements can multiply significantly.

For example, consider a disk storage (SC A) with two hierarchies:

- **Hierarchy 1**: Disk (SC A): Single Copy, Non-Striped (SC B)
  - 10 Total Migration Streams, Stripe Width 1
  - Maximum tape mounts: 10 (10 per copy level)
- **Hierarchy 2**: Disk (SC A): Dual Copy, Non-Striped (SC C and SC D)
  - 10 Total Migration Streams, Stripe Width 1, 2 Copy Levels
  - Maximum tape mounts: 20 (10 per copy level)



If both hierarchies have eligible files for migration at the same time, they will migrate in parallel, resulting in up to **30 simultaneous tape mounts** (10 + 20).

This parallel behavior allows for flexible data management policies but requires careful resource planning. Sites should consider:

- The total number of available tape drives
- The likelihood of concurrent migration across multiple hierarchies
- Whether different hierarchies can share tape drives or if dedicated drives are needed
- The impact on other HPSS operations (recalls, user writes, etc.)

When planning tape drive capacity, calculate the worst-case scenario where all hierarchies from a disk storage class are migrating simultaneously, then ensure sufficient drives are avaialble for both migration and normal operations.

Total Migration Streams

The total number of parallel migration streams that will be run during migration for each copy level in the hierarchy. This value must be a multiple of the **Number of Migration Streams Per File Family**. The **Total Migration Streams** value determines the maximum number of migration streams that can be active simultaneously across all file families for a given copy level. For example, if **Total Migration Streams** is set to 10 and **Number of Migration Streams Per File Family** is set to 2, then up to 5 file families can be migrated in parallel (10 total streams ÷ 2 streams per family = 5 families).



Advice

*Setting the **Total Migration Streams** value too high may result in higher than desired tape drive utilization. Each hierarchy within a disk storage class is migrated in a separate but parallel fashion. As such, when there are files eligible for migration in each hierarchy within a disk storage class, tape mount requests will be generated for each hierarchy. The **Total Migration Streams** value may be used indirectly to control how many tape mounts are requested, and thus how many drives are used during migration. The maximum number of drives needed for a migration can be calculated as the total stripe width value (**SW**) multiplied by the **Total Migration Streams** value (**TMS**) for every hierarchy and disk-to-tape copy level within the storage class:*

$$(SW \times TMS)_{Hier1,LevelA} + (SW \times TMS)_{Hier1,LevelB} + \cdots$$

$$(SW \times TMS)_{Hier2,LevelA} + (SW \times TMS)_{Hier2,LevelB} + \cdots$$

$$+ \cdots +$$

$$(SW \times TMS)_{Hiern,Levelx} + (SW \times TMS)_{Hiern,Levely} + \cdots$$

For example, an HPSS system that has a disk storage class which migrates to a tape storage class with total stripe width two. The migration policy's stream count is also two. In this situation, the maximum tape drive usage would be four: two sets of two-wide tape VVs.

A hierarchy that migrates data from disk to a first and second 4-wide tape copy with a stream count of one would use a maximum of eight tape drives: the first copy would go to one set of 4-wide tapes and the second copy would migrate to a second set of 4-wide tapes. A second hierarchy added to the mix with the same characteristics would double the number of tape drives needed.

Now consider an HPSS system that has disk storage class A at the top level of hierarchy 1 and hierarchy 2. Storage class A has a migration policy with a total stream count of 4. Hierarchy 1 migrates to a tape storage class with total stripe width of two. Hierarchy 2 migrates to a first copy tape storage class with total stripe width of two, and a second copy tape storage class with a stripe width of one. The total drive utilization for a maximal migration case (when there exist files eligible for migration in all hierarchies of the disk storage class) will be twenty:

$$(2_{SW} \times 4_{TMS})_{Hier1,Level1} + (2_{SW} \times 4_{TMS})_{Hier2,Level1} + (1_{SW} \times 4_{TMS})_{Hier2,Level2} = 20$$

*In addition to the **Total Migration Streams**, the **Maximum VVs to Write** parameter in the Tape Storage Class Configuration window may be used to further control how many tape drives/cartridges are used for migration: it will place a cap on the maximum number of VVs that can be written simultaneously in a storage class.*



See Migration Stream Distinction for additional information.

Triggered Migration Options

There are three choices for managing migration behavior when a storage class is running out of space and the next migration isn't yet scheduled to occur:

Migrate when Warning Threshold Exceeded

A migration run should be started immediately when the storage class warning threshold is exceeded.

Migrate when Critical Threshold Exceeded

A migration run should be started immediately when the storage class critical threshold is exceeded.

Migrate only at Scheduled Intervals

Do not trigger unscheduled migrations when storage class thresholds are exceeded.

Optimize for Even File Family Migration

Select this option in order to evenly migrate files from all file families with migratable files. Otherwise, the MPS will migrate all files within a file family before moving onward to the next file family.

Read-Optimized Aggregates

Select this option if you want aggregates optimized for read. **Read-Optimized Aggregates** will have only one ordering key value per aggregate (e.g. only files contained in Directory X). When **Read-Optimized Aggregates** is not selected, you will get Write-Optimized Aggregates which allows more than one ordering key value per aggregate (e.g. files contained in Directory X followed by files contained in Directory Y, etc). The default behavior is Write-Optimized Aggregates. This is another important aspect to consider when preparing to employ Full Aggregate Recall.

Max Aggregate Size

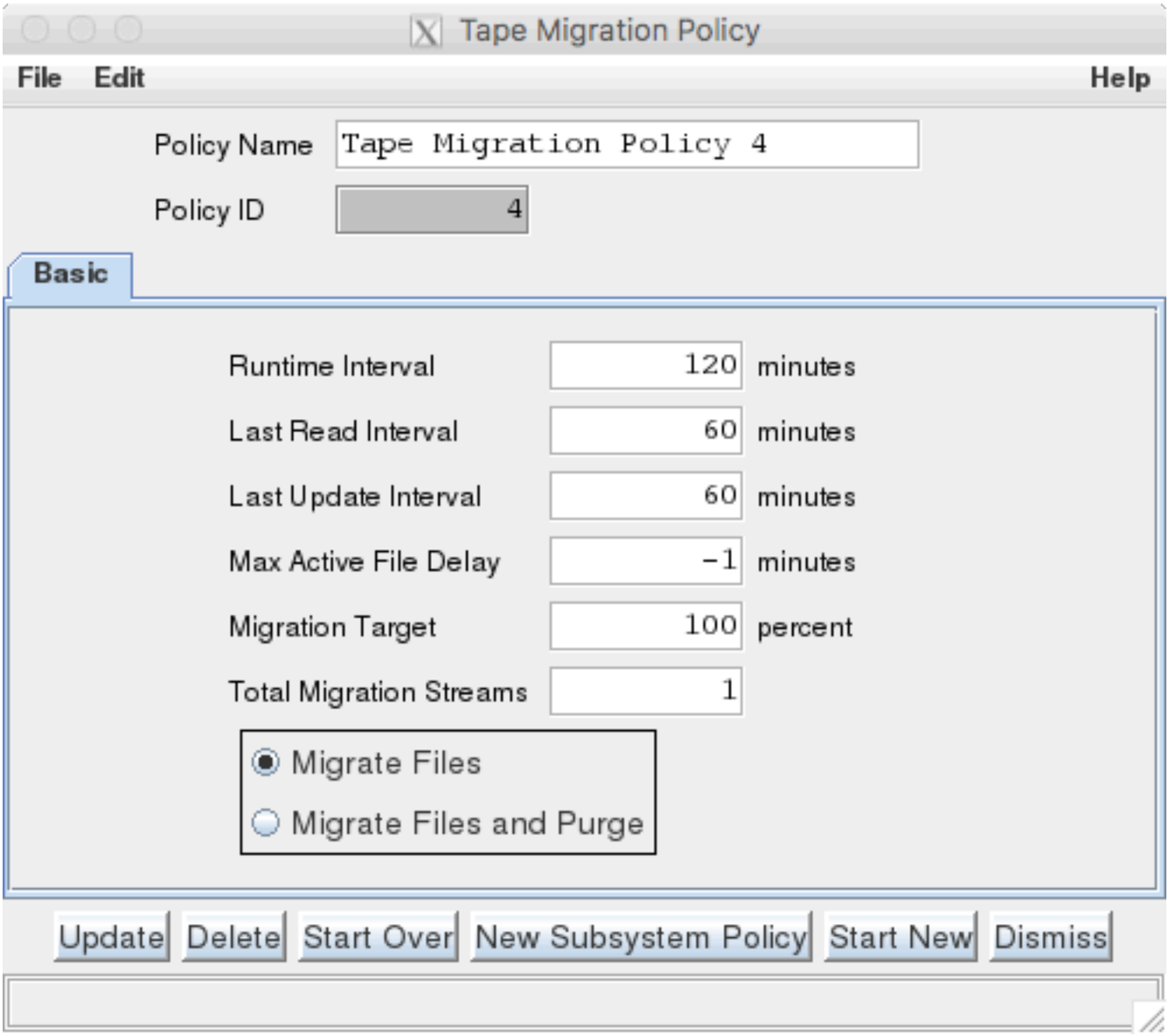
The maximum total size in bytes of a tape aggregate. Disk-to-tape migration bundles bitfiles into tape aggregates, and this value sets the maximum aggregate size. Tape aggregates are limited to a maximum of the total estimated size of the destination tape volume. Tape aggregation can be disabled by setting the "Max Aggregate Size" to 0.



Advice

For details about migration of files to aggregates on tape, see Overview of tape aggregation.

17.4.2.2. Tape Migration Policy configuration



This window allows an administrator to manage tape migration policies and their subsystem-specific overrides.

Subsystem-specific policies define migration rules to be applied on a subsystem basis instead of using the default (basic) migration policy. When a migration run occurs and the storage class's migration policy does not have a subsystem-specific migration policy with the same subsystem ID, then the basic migration policy will be applied; otherwise, the matching subsystem-specific migration policy will be used. Click the **Create Tape** button on the *Migration Policies* window to create a new policy. To bring up the configuration for an existing policy, select the policy from the list and click on the **Configure** button.

Field descriptions

Policy Name

The descriptive name of the migration policy.

Policy ID

A unique ID associated with the migration policy.

Runtime Interval

The number of minutes to wait between the end of one migration run and the start of the next. This can be in the range [0 to 4,294,967,295].

Last Read Interval

The number of minutes that must pass since a file or tape volume was last read before it can become a candidate for migration.

Last Update Interval

The number of minutes that must pass since a file or tape volume was last written before it can become a candidate for migration.

Max Active File Delay

The maximum number of minutes beyond the larger of the Last Read and Last Update intervals to wait for the intervals to be satisfied before forcing the migration of files. This can be in the range [-1 to 2,147,483,647].

Example 1. Max Active File Delay values

A value of "-1" means wait to indefinitely for the intervals to be satisfied. A value of "0" means to wait no longer than the larger of the two intervals. A value of "2880" means to wait no more than two days beyond the larger of the two intervals. If a file or tape were to remain busy for two days beyond the initial Last Read and Last Update intervals, the MPS would force the migration of the delayed files even if the files or source tape volume were active.

Migration Target

The desired percentage of free tape space to have available when migration is completed. The migration will be terminated when this goal is reached or when no more files meet the migration criteria. This goal may not be attainable if the total size of all files not eligible for migration is large.

Total Migration Streams

This value determines the degree of parallelism in the file migration process. This applies to policies using the **Migrate Files** and **Migrate Files and Purge** options only (see **File Options** below).

File Options

There are two options available for determining how tape migration handles files. Only one of these options may be selected at a time.

Migrate Files

A duplicate copy of the tape files will be made at the next lower level in the hierarchy.

This enables a tape migration algorithm which is similar to disk migration. This algorithm is based on individual files rather than tape volumes, and is able to make second copies of files stored on tape. In this algorithm, individual files are selected for migration based on their last

write time and the settings in the migration policy. The selected files are migrated downwards to the next level in the hierarchy. The order in which files are migrated is based approximately on the order in which they are written to tape. Prior to migration, however, the files are sorted by source tape volume, so their migration order may vary. In this algorithm, files at the source level are never purged. Note that the source tape storage class must be repacked and reclaimed manually.

Migrate Files and Purge

The tape files will be moved to the next lower level in the hierarchy. No duplicate copy is maintained.

This enables the same file-based tape migration algorithm as **Migrate Files**, but purges files from the source level when they are successfully migrated. Additionally, this algorithm considers both the read and write times for a candidate file against the settings in the migration policy. This algorithm is useful for moving inactive files to a lower hierarchy level. Note that the source tape storage class must be repacked and reclaimed manually. Also note that if migrated files are staged up in the hierarchy, HPSS does not run a purge policy to delete the cached copy at the higher level.

Related information

Migration and purge policy overrides

17.4.2.3. Changing a migration policy

To update an existing basic migration policy, select the policy from the *Migration Policies* list window and click the **Configure** button. The disk or tape *Migration Policy* window will appear. Make the desired changes to the basic policy and subsystem policies by selecting the appropriate tabs and placing new values in fields. When all changes have been made, click the **Update** button to record the changes.

In order for changes made to a migration policy to take effect, the changes should be applied by clicking the **Apply Config** button on the *HPSS Health and Status* window. This will ensure the Migration/Purge servers and the Core servers are reinitialized and that the changes to the migration policy are applied to all storage classes that reference the policy. You can also reread the policy from the *MPS Storage Class Information* window or the **Migration Controls** selection list on the *Active Storage Classes* list screen. However, the changes are only applied to the storage class and storage subsystem for which the policy is reread.

17.4.2.4. Deleting a migration policy



*Before deleting a basic migration policy, make sure that it is not referenced in any storage class configuration. Users cannot delete a migration policy that is in use by a storage class. Attempts to delete a migration policy that is referenced will fail due to database referential integrity constraints. Be sure to check the **Migr Policy** column of the Configured Storage Classes list for any storage class that references the migration policy to be deleted.*

*Be sure to click the **Apply Config** button on the HPSS Health and Status window*

after deleting a migration policy.

To delete a migration policy, select the policy from the *Migration Policies* list window and click the **Delete** button. If a basic policy is selected, and the policy has subsystem-specific policies associated with it, a prompt will appear asking if the basic policy and the related subsystem-specific policies should all be deleted since they must be deleted before the basic policy can be.

A migration policy may also be deleted by clicking the **Delete** button on the *Migration Policy* configuration window for the policy. Subsystem-specific policies can be individually deleted from the *Migration Policy* configuration window for the policy by selecting the specific subsystem's tab and clicking the **Delete Subsystem Policy** button.

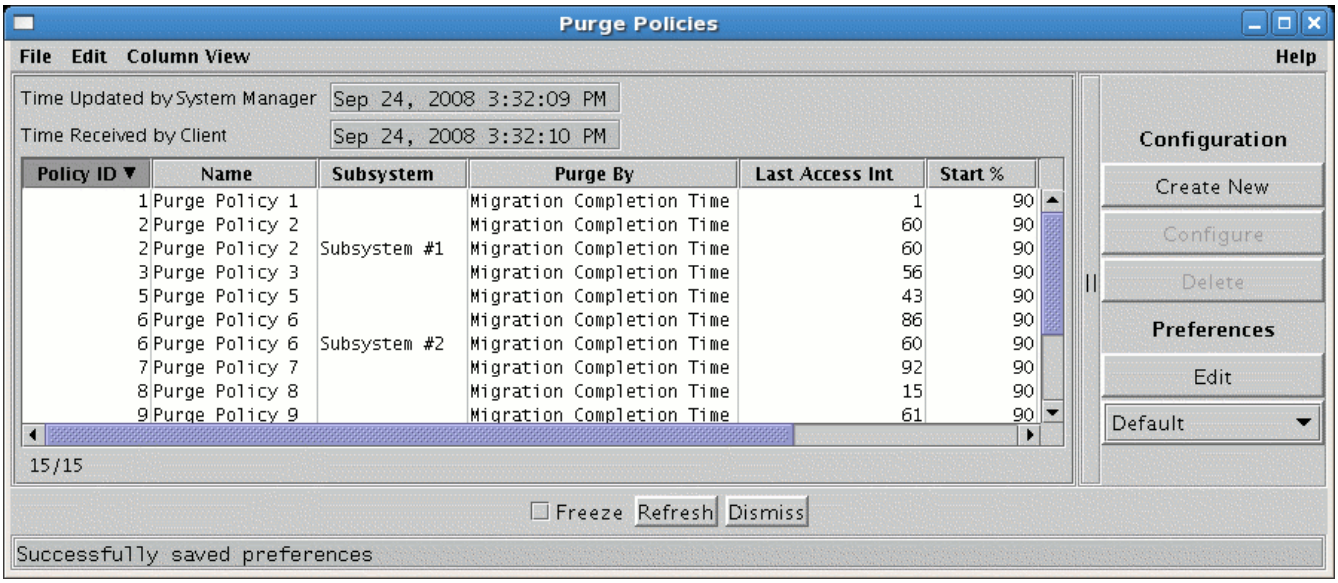
17.5. Purge policies

A purge policy is associated with a disk storage class and defines the criteria by which data is purged from that storage class once migration has copied that data to storage classes at lower levels in the storage hierarchies.

A basic purge policy must be assigned to all disk storage classes that require purge services. This basic policy determines the purge parameters for the storage class across all storage subsystems. If individual purge policies are desired for specific storage subsystems, storage subsystem-specific purge policies may be added which override the default values in the basic policy. The storage subsystem-specific purge policies share the same purge Policy ID and Policy Name with the basic policy.

Both basic and subsystem-specific purge policies are created and managed using the *Purge Policies* window. The basic policy must be created before creating the subsystem-specific policies.

17.5.1. Purge Policies window



This window displays a list of all the purge policies configured in the system.

To create a new purge policy, click the **Create New** button in the *Purge Policies* window. A new *Purge Policies* window appears. Enter the desired parameters for the policy, and then click the **Add**

button. The policy will be added to the system. If subsystem-specific policies are desired, click the **New Subsystem Policy** button after the window refreshes, enter the specific purge policy parameters, and click the **Update** button. This process can be repeated for each subsystem.



*When a purge policy is added to an existing storage class, the Migration/Purge Servers for that storage class and the Core Servers must be reinitialized or restarted in order for the policy to take effect. It is recommended to use the **Apply Config** button on the HPSS Health and Status window to reinitialize the servers.*

Field descriptions

Purge Policy List columns

The columns provide the same information that can be found on the *Purge Policies* window in the following section.

Configuration buttons

Create New

Open a *Purge Policies* window with default values for a new policy.

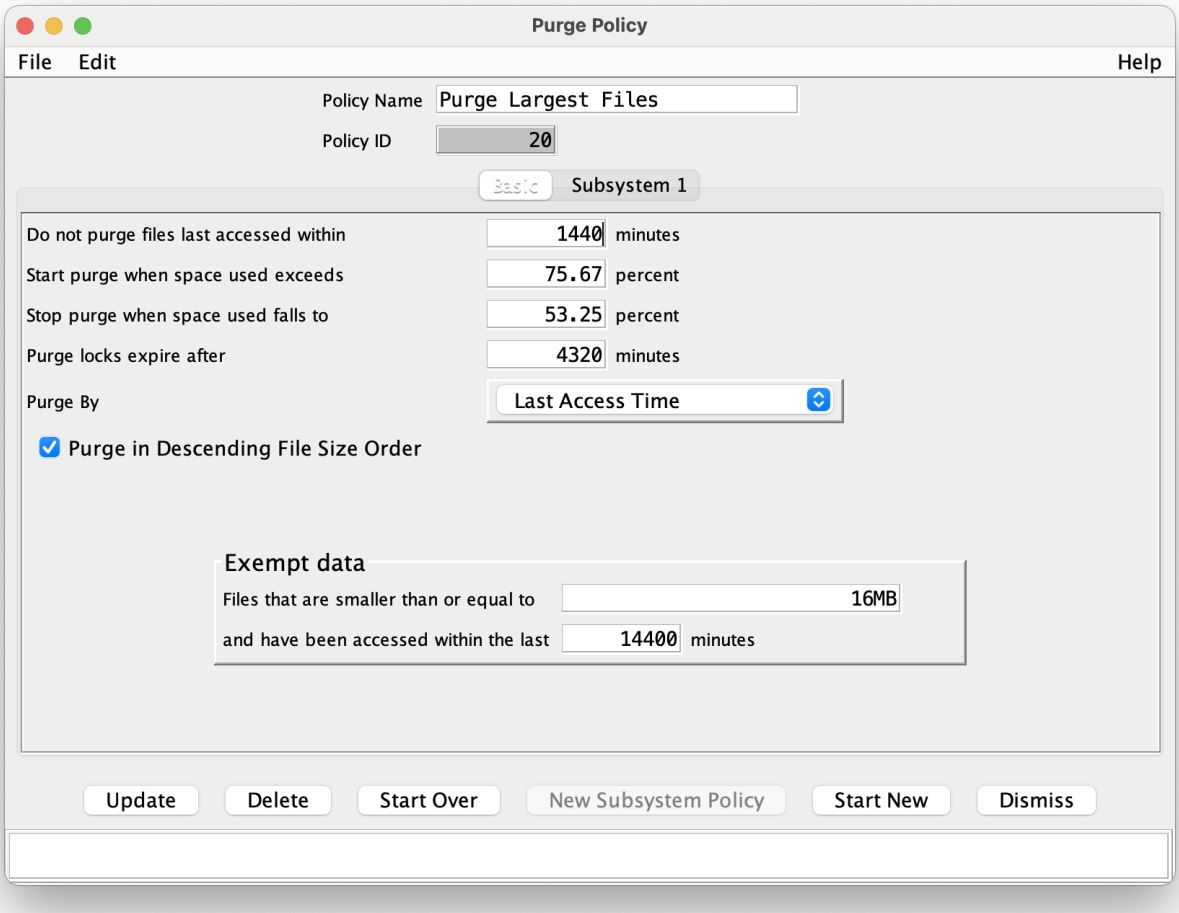
Configure

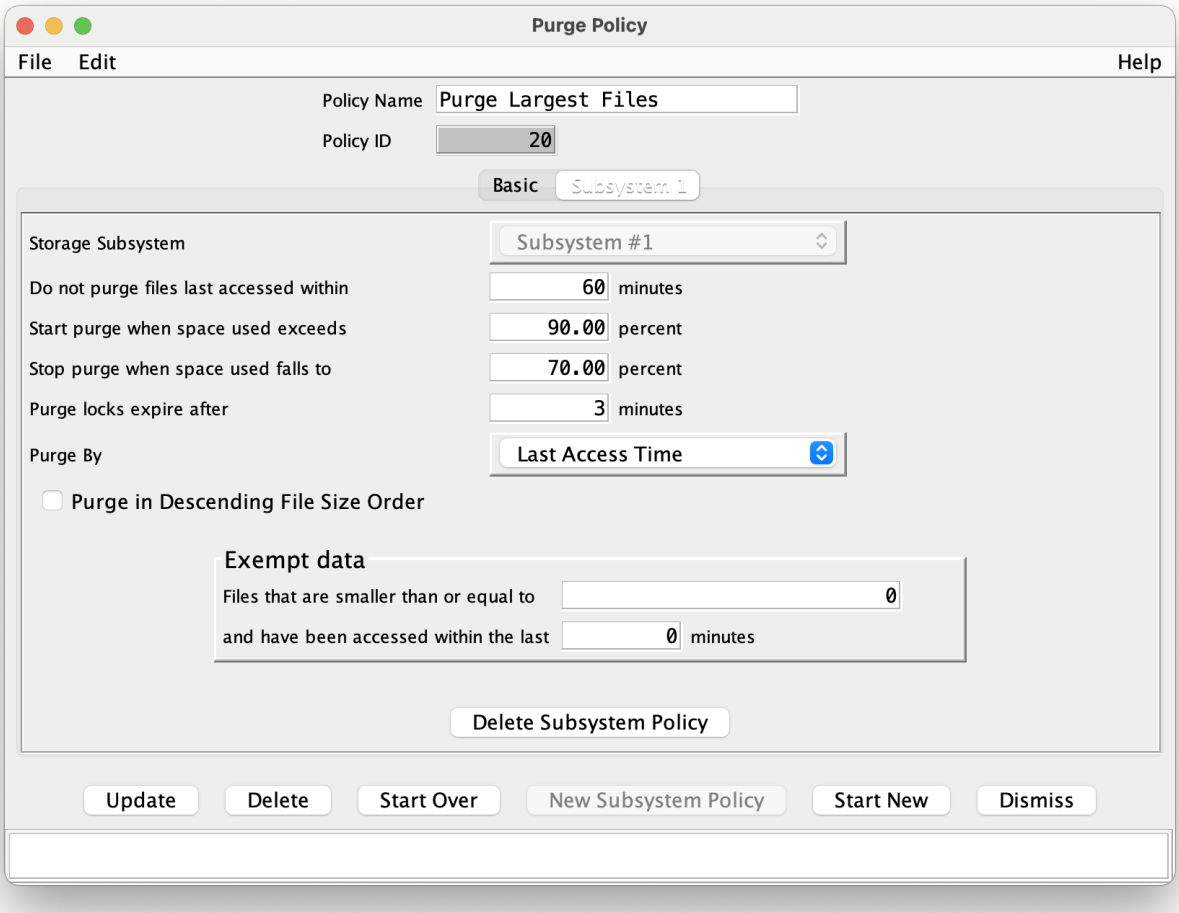
Open the one or more selected purge policies for editing.

Delete

Delete the one or more selected purge policies.

17.5.2. Purge Policy configuration


Purge Policy
File
Edit
Help
Policy Name
Purge Largest Files
Policy ID
20
Basic
Subsystem 1
Do not purge files last accessed within
1440
minutes
Start purge when space used exceeds
75.67
percent
Stop purge when space used falls to
53.25
percent
Purge locks expire after
4320
minutes
Purge By
Last Access Time
Purge in Descending File Size Order
Exempt data
Files that are smaller than or equal to
16MB
and have been accessed within the last
14400
minutes
Update
Delete
Start Over
New Subsystem Policy
Start New
Dismiss



This window allows you to manage a purge policy. Purge policies are assigned to storage classes to tell the Migration/Purge Server and Core Server how to free disk space occupied by files which have been migrated. Purge policies apply to disk storage classes only.

The window always includes the **Basic** tab and may include one or more **Subsystem** tabs. These will be referred to as "basic" and "subsystem" below.

Each purge policy consists of a single basic policy and zero or more subsystem policies. Each subsystem policy has the same policy ID and name as its basic policy. When a new purge policy is needed, the basic policy must be created first; subsystem policies can then be created as needed.

When HPSS needs to apply a purge policy to a subsystem, it first searches for a matching subsystem policy. If none is found, HPSS uses the basic policy. You can consider the basic policy to be the default for all subsystems which do not have policies of their own.

After changing any purge policy information, the changes will take effect after clicking the **Apply Config** button on the *HPSS Health and Status* screen.

When creating a subsystem policy, fields which differ from the basic policy values can be thought of as exceptions to the basic policy values. However, this does not imply that unchanged defaults maintain any links with the basic policy. Changes in the basic policy have no effect on subsystem policies.

Field descriptions

Policy Name

The descriptive name of the Purge Policy.

Policy ID

A unique ID associated with the Purge Policy.

Do not purge files last accessed within

A file will not be a candidate for purge until it has remained unaccessed (for read or write) for the length of time specified by this field.

Start purge when space used exceeds

Purge will begin for a storage class when the amount of its space used exceeds this threshold. Used space includes any file in the storage class, whether it has been migrated or not. This threshold may be expressed as a fractional percentage with a precision of up to two decimal places (for example, `85.42` percent).

Stop purge when space used falls to

Purging will stop for a storage class when the amount of its space used drops to this threshold. Note that the purge may stop before this point if it runs out of files which are available for purging. This threshold may be expressed as a fractional percentage with a precision of up to two decimal places (for example, `80.24` percent).

Purge Locks expire after

Maximum number of minutes that a file may hold a purge lock. Purge locked files are not eligible for purging. A value of "0" indicates that purge locks expire immediately.

Files may be purge locked to the highest level (level zero) of a hierarchy, provided that level zero is of type disk, and data exists at level zero. Once a file is purge locked, it is no longer a purge candidate. By entering a "lock expiration duration," you will be able to monitor the number of purge locked files and the number of expired purge locked files using the Migration/Purge Server (MPS) log. After a purge run, the MPS will log the total number of purge locked files and the number of expired purge locks (those files that have been locked longer than this specified number of minutes). To view the names of the files with expired purge locks, use the Purge List Utility (plu(7)) with the **-x** parameter on the given storage subsystem.

Super purge locks are not configurable by the "lock expiration duration." They do not expire and they are not yet reported by the MPS or the plu(7) utility. Super purge locks should be rare. They are set temporarily by the recover(7) utility and released when the recover(7) utility determines that a safe copy of the file still exists. They are set internally by the Core Server whenever a manual force purge is performed on a file, which should not be done except with guidance from HPSS support. Super purge locks may be removed manually by the **scrub purgeunlock** command with the assistance of HPSS support.

Purge by

The MPS uses this time attribute of a file to determine which files are eligible to be purged.

By default, files are selected for purge based on the time the purge record was created, the Migration Completion Time. Alternately, the selection of files for purging may be based on the

time the file was created, the File Creation Time, or the time the file was last accessed, the Last Access Time.



*If this field is changed, use the **Apply Config** button on the HPSS Health and Status screen to apply the change across the Core Servers and Migration/Purge Servers. Additionally, files will be purged in an unpredictable order until all purge records existing at the time of the change are cleared.*

Purge in Descending File Size Order

When checked, the purge candidate list will be retrieved by file size in descending order. Files will be purged in order by file size with the largest files purged first.

Exempt data

The exempt data includes two conditions to prevent certain files from being identified as purge candidates. Both conditions must be true in order for a file to be exempt as a purge candidate. Otherwise, the file will be subject to the standard purge policy.

Files that are smaller than

A file that is the specified size or smaller in bytes may be ineligible as a purge candidate.

and have been accessed within the last

A file may not be a candidate for purge until the time elapsed for the most recent read or write time exceeds the value of this field.

Storage Subsystem (subsystem policy tab only)

The descriptive name of the storage subsystem to which a subsystem-specific policy applies. This field is filled in with the selected storage subsystem name at the time a subsystem-specific policy is created and may not be changed afterwards.

Related information

Migration and purge policy overrides

17.5.3. Changing a purge policy

To update an existing basic purge policy, select the policy from the *Purge Policies* window and click the **Configure** button. After modifying the basic policy, click on the **Update** button to save the changes. To update an existing subsystem-specific purge policy, first select and display the existing basic policy and then select the tab for the desired subsystem policy. After modifying the subsystem-specific policy, click on the **Update** button to save the changes.

In order for changes made to a purge policy to take effect, the changes should be applied by clicking the **Apply Config** button on the *HPSS Health and Status* window. This will ensure the Migration/Purge servers and the Core servers are reinitialized and that the changes to the purge policy are applied to all storage classes that reference the policy. You can also reread the policy from the *MPS Storage Class Information* window or the **Purge Controls** selection list on the *Active Storage Classes* list screen. However, the changes are only applied to the storage class and storage subsystem for which the policy is reread.

17.5.4. Deleting a purge policy



Before deleting a basic purge policy, make sure that it is not referenced in any storage class configuration. Users cannot delete a purge policy that is in use by another storage class. Attempts to delete a purge policy that is referenced will fail due to database referential integrity constraints. Be sure to check the Purge Policy column of the Configured Storage Classes list for any storage class that references the purge policy to be deleted.

*Be sure to click the **Apply Config** button on the HPSS Health and Status window after deleting a purge policy.*

To delete a purge policy, select the policy from the *Purge Policies* window and click the **Delete** button. If a basic policy is selected, and the policy has subsystem-specific policies associated with it, a prompt will appear asking if the basic policy and the related subsystem-specific policies should all be deleted since the basic policy cannot be deleted until all of the subsystem-specific policies are deleted.

A purge policy may also be deleted by clicking the **Delete** button on the *Purge Policy* configuration window for the policy. Subsystem-specific policies can be individually deleted from the *Purge Policy* configuration window for the policy by selecting the specific subsystem's tab and clicking the **Delete Subsystem Policy** button.

17.5.5. Purge parallelism

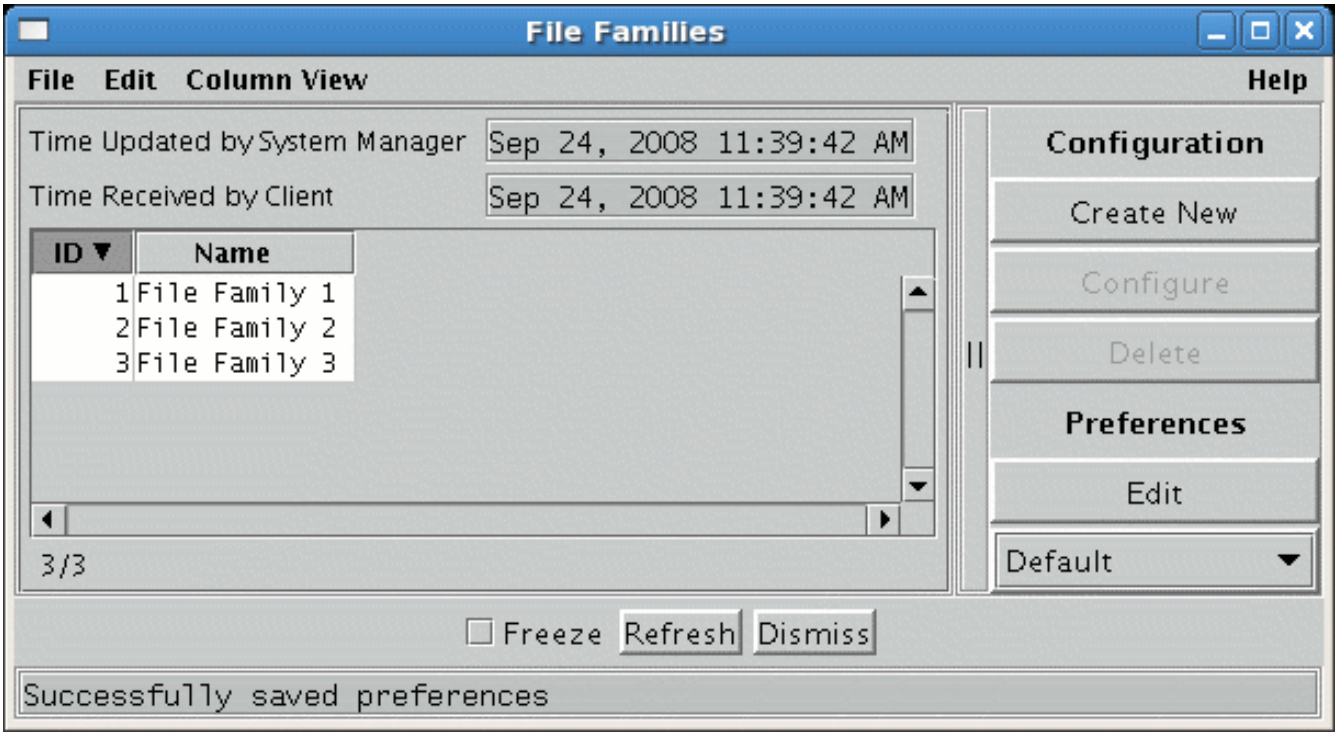
The degree of parallelism used for purging files from disk storage classes can be controlled via the HPSS environment. By default, the MPS degree of parallelism is set to two threads per storage class. You *may* find that increasing the number of purge threads increases the speed at which your HPSS system can purge files from disk. An increase in the number of purge threads will cause a slight increase in the amount of memory used by the MPS.

To run with a custom number of purge threads, use the HPSS environment variable named HPSS_MPS_PURGE_PARALLELISM. The valid range of values for this variable is [1,10]. As an example, for the MPS to run with four threads of purge per storage class, the following needs to be specified in `<HPSS_PATH_ETC>/env.conf`:

```
HPSS_MPS_PURGE_PARALLELISM=4
```

After changing the degree of purge parallelism, restart the MPS for it to begin using the new configuration.

17.6. File families



This window lists the configured file families.

Field descriptions

The fields of the columns of this window are those of the file family configuration described in File Family Configuration.

Configuration buttons

Create New

Open a *File Family Configuration* window with default values.

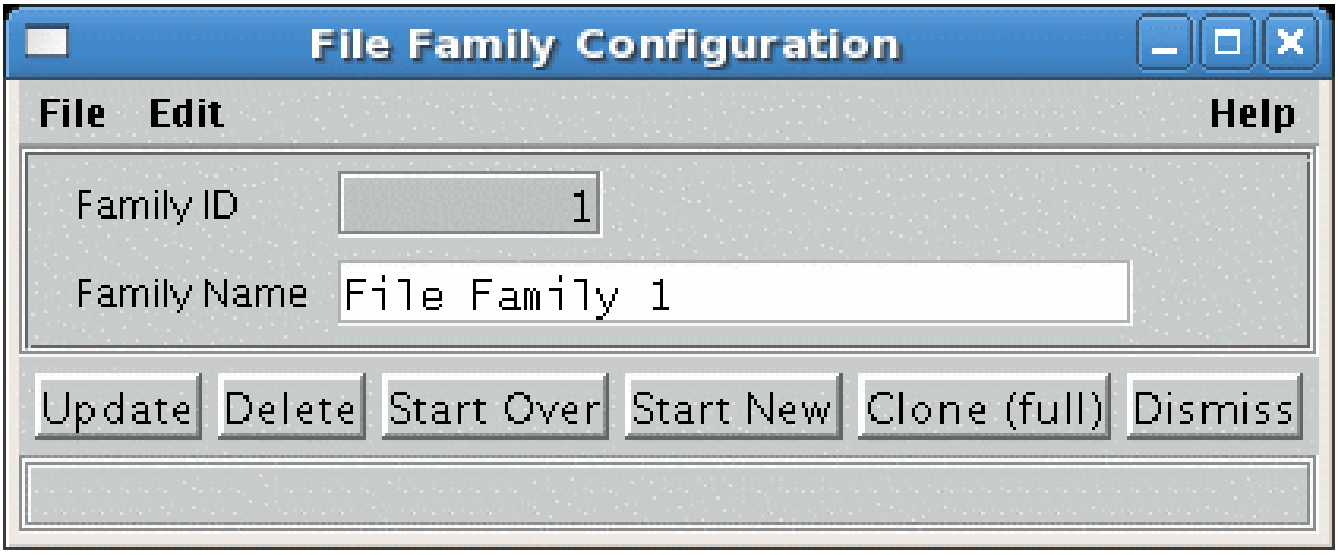
Configure

Open the selected file family configurations for editing.

Delete

Delete the selected file family configurations.

17.6.1. File Family Configuration



This window allows you to manage a file family configuration.

Field descriptions

Family ID

A positive integer which serves as a unique identifier for this file family.

Family Name

The descriptive name for this file family. The name should be unique among all file families in this HPSS installation. The default value is "File Family <ID>".

17.6.2. Changing a file family

The only change that can be made to a file family is to change the **Family Name**. This action has no side effects and takes effect when the **Update** button is clicked.

17.6.3. Deleting a file family

SSM provides a means to delete a file family definition, but this action has no effect on files already recorded in the family. The file family attribute is permanent and cannot be removed or changed in existing files. Deleting a file family just means that files can no longer be added to the deleted file family.

Chapter 18. Device and drive management

Every disk,tape and object store drive that is used by HPSS is controlled by two servers. The PVL performs mounts and dismounts (for disk devices these are logical operations only), and the Movers perform I/O. In support of these two views, the terms "PVL drive" and "Mover device" are used to refer to the configuration and information maintained about the drive by the PVL and Mover, respectively.

All configuration information is managed through a single SSM window. However, there are separate PVL drive and Mover device information windows, each with their own administrative state and other settings. When administrative changes are made, it is important to know whether they are to be performed against the PVL drive, the Mover device, or both.

Devices and drives are added and deleted dynamically. This means that they are added and deleted while the system is up. The PVL must be running and it is expected that the associated Mover and PVR (for tape) are also running. A new device and drive cannot be used until the PVL has successfully informed the associated Mover and PVR about the new device and drive. Likewise, a previously deleted device and drive cannot be added back until the PVL has successfully informed the associated Mover and PVR about the delete, and a previously added device and drive cannot be deleted until the PVL has successfully informed the associated Mover and PVR about the add. Additional details are described later in this chapter.

Devices and drives can be updated dynamically (that is, while the PVL, associated Mover, and associated PVR are running) via configuration update. The drive (tape) must be locked and idle before an update can occur. Additionally for disk drives, the volume must be unallocated before an update can occur. However this restriction can be omitted for some drive attributes for which the configuration metadata can be updated dynamically via the *PVL Drive Information* window. Similarly there are some device attributes for which the server "in memory" values can be changed dynamically (but not preserved when the Mover is restarted) via the *Mover Device Information* window. Additional details are described later in this chapter.

Note that all newly-created PVL drives are locked (that is, the *Devices and Drives* window **Drive Admin State** reports "Locked"), which prevents mounts from occurring on the drive. Therefore, these drives will need to be unlocked before they can be used. This can be done by highlighting the device/drive in the *Devices and Drives* window and clicking the "Unlock" button under the **Drive Administration** group heading.

Review the appropriate subsections of Physical Volume Repository (PVR)-specific configuration before configuring a device/drive or modifying its state.

18.1. Configure a new device and drive

Be certain all tape drives are configured to use variable block sizes. See Enable variable block sizes for tape devices. These steps are also covered in detail in the Tape devices section.

The Device and Drive configuration entry can be created and managed using the *Devices and Drives* window.

After a drive has been configured, its state can be changed (unlocked versus locked) using the state change instructions described in the *Devices and Drives* window section.

For disks using a file system interface, sparse files need to be created. The UNIX **truncate** utility is a good way to do this. The file name of the file will go in the **Device Name** field when configuring the device. Some file systems have features (such as software RAID or erasure coding) that can be used to enhance the integrity of the data stored on them, or they have features (such as compression) that increase the capacity of the storage media. These features can be used by the file system, but do not affect HPSS processing.

Before adding a tape drive to HPSS, the administrator must first fully configure the tape hardware and driver software into the host platform and operating system. For disk drives, the raw disk must first be defined in the operating system. Note that using the block special file name in the device configuration may result in a large performance degradation. For SAN disk devices, the hpss_san3p_part(7) utility with the **-i** option and device name must be run to assign a UUID to the disk device. This UUID should be used in the **Device Name** field when configuring the disk device.

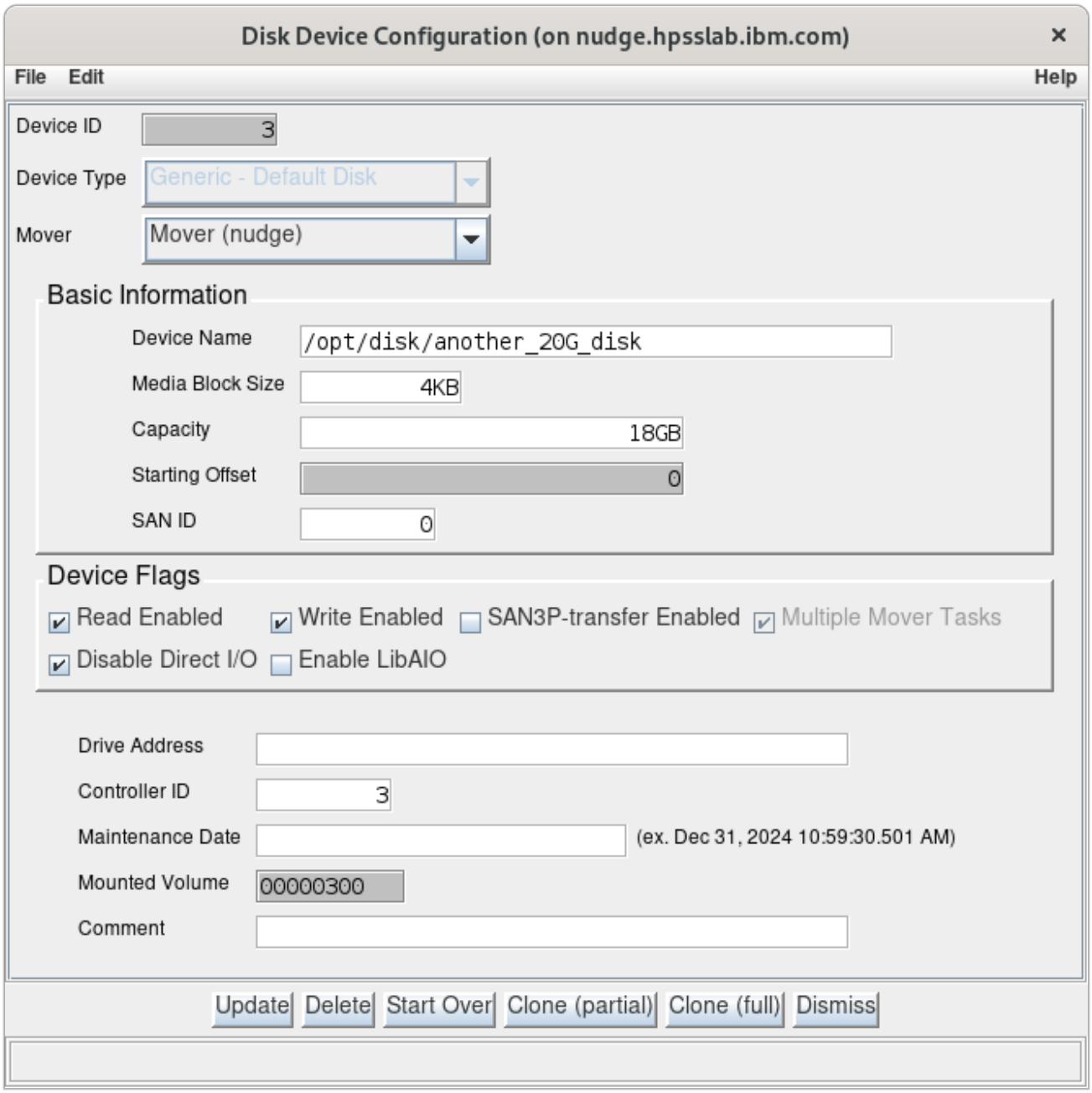
Currently the number of drives which may be configured per PVR is limited to 256. The current maximum number of PVRs is 64. The maximum number of devices per Mover is also 64. Given these limits, the total number of tape drives allowed in the system is 16,384.

Devices and drives are added dynamically. This means that they must be added while the system is running. The PVL must be running and it is expected that the associated Mover and associated PVR (for tape) are also running. A new device/drive cannot be used until the PVL has successfully notified the associated Mover and PVR (for tape) about the new device/drive. If the PVL is unable to successfully notify the Mover or PVR about the new device/drive, it will retry several times; each failure will be logged as an ALARM. After the last failed attempt to notify the Mover or PVR, the PVL will abandon retrying. The administrator is advised to stop and start the Mover (if it was up during the failed retries) and to reinitialize the associated PVR (if it was up during the failed retries) before the device/drive can be used.

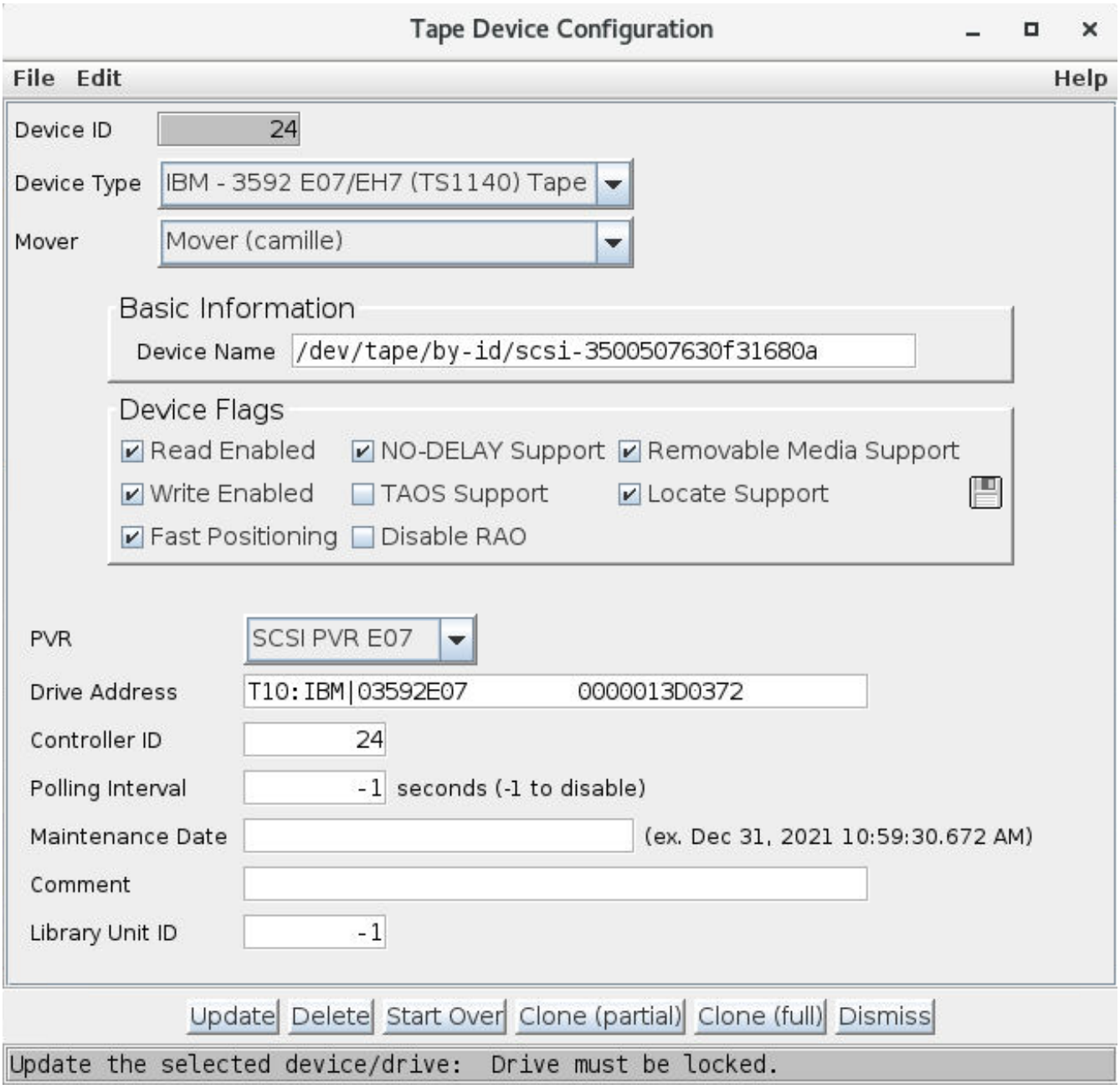
Likewise, a previously deleted device/drive cannot be added back until the PVL has successfully notified the associated Mover and PVR (for tape) about the previous delete. If the administrator attempts to add a device/drive with a previously deleted Drive ID that is still pending delete notification, the administrator will get a BUSY failure and the pending notification will be aborted. An ALARM will be logged informing the administrator that the create failed since the Mover/PVR notification pends, and another ALARM will be logged informing the administrator that the previous drive delete notification will be aborted and to try later. Wait a minute or so before retrying in order to give the PVL time to abort the notification. The **Drive Flag** field in the *Devices and Drives* window and *PVL Drive Information* window will report whether or not a particular drive is pending Mover/PVR notification. Of course, this is only viewable while the drive exists.

At this point, the Administrative State of the drive must be unlocked before it is available. Refer to Unlocking a drive.

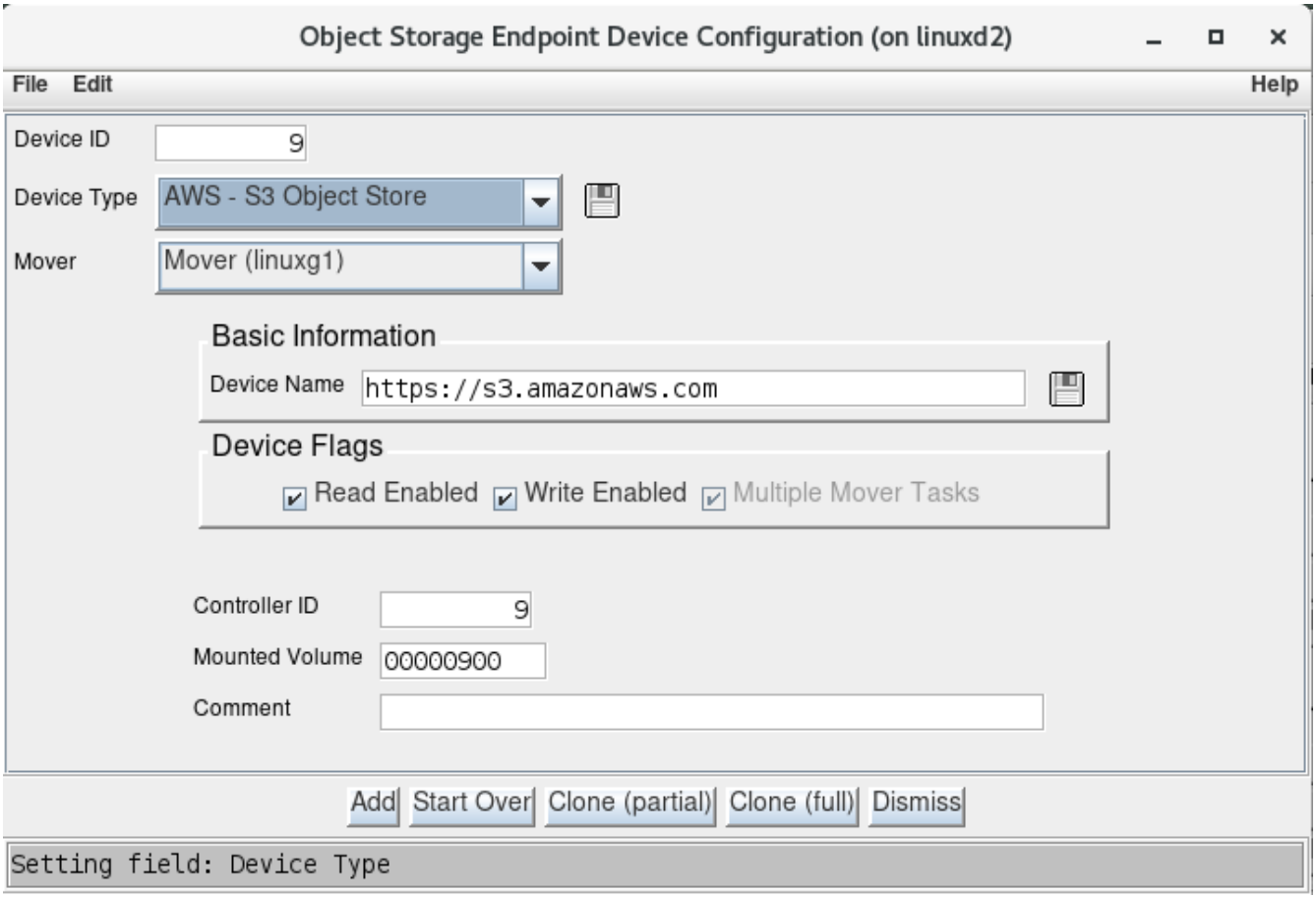
Disk Device Configuration



Tape Device Configuration



Object Store Device Configuration



These windows allow you to manage a tape, object store, or disk device/drive configuration.

Modifying/Updating the device/drive configuration via the *Tape Device Configuration*, *Disk Device Configuration*, or *Object Store Device Configuration* windows is managed dynamically by the PVL. The associated Movers, or the associated PVRs (for tape) are expected to be running. If you are changing the Mover and/or PVR, then both the current Mover/PVR and the new Mover/PVR will need to be running. An option exists which allows the PVL to perform the update without associated servers running. This option should be used with caution as it mainly addresses the abnormal condition where a Mover platform has failed and drives are being moved to a new platform/Mover and should be used with caution.

Additionally PVL drive configuration metadata updates can occur dynamically via the *PVL Drive Information* window. Likewise, some Mover device configuration "in memory" updates (that is, metadata not changed) can occur dynamically via the *Mover Device Information* window. Refer to Changing a drive's configuration.

Field descriptions

Device ID

The ID associated with this device/drive. Any positive 32-bit integer value.

Device Type

The type of device over which data will move.



Advice

The AWS - S3 Object Store is currently the only supported object store device.

Mover

The name of the Mover that controls the device.



Advice

A single Mover can manage a maximum of 64 devices. When the Mover is started and more than 64 devices are found, it will log an error and exit.

Device Name

The name by which the Mover can access the device. This name is usually the path name of a device special file such as `/dev/rmt0`. For SAN disks, this name is instead a UUID assigned to the disk device by running the hpss_san3p_part(7) utility with the **-i** option and device name. The UUID for a SAN disk device can be determined by running the hpss_san3p_part(7) utility with the **-s** option.

Advice

For locally attached disk devices, the pathname should refer to the raw/character special file (for example, `/dev/rhpss_disk1`).

For Linux systems, raw disk devices are deprecated so HPSS will support block devices and block device partitions. The SCSI disk block devices are referred to by pathnames of the form `/dev/sdXY`, where X begins at "a" and is incremented for each LUN detected and Y begins at "0" and is incremented for each partition of the block device. See sfdisk(8) for details about partitioning Linux block devices. When a block device is specified, the Mover will use O_DIRECT when opening the device and thus bypass the normal buffering supported by Linux block devices. SCSI tape devices are referred to by pathnames of the form `/dev/stX`, where X begins at zero and is incremented for each LUN detected.



*For SAN3P devices on Linux, Linux DM Multipath can be used instead of the default SCSI block devices. This is done by setting the HPSS_SAN3P_LINUX_DMMPATH environment to "on". The environment setting can be added with the **hpss_env_set** command, or by simply using your favorite editor to change the `/var/hpss/etc/env.conf` file. The setting should be added for each Mover and client that needs to access a SAN3P device. If you have previously been using a SAN3P configuration, deletion of the `/var/tmp/hpss_san3p.id.*` files on each client and Mover may be required. These files will be regenerated for the next SAN3P access.*

*For disk storage technologies that are using a file system interface, use the file name of the sparse file. In addition, when filling in the **Capacity** field, use the size of the sparse file.*

For object store devices, this is the base URL of the endpoint supporting access to the object store.

Media Block Size (disk only)

The block size for the device. This value should be a power of two multiple and a multiple of the underlying disk block size; otherwise, an error will occur at first I/O.

Capacity (disk only)

The size of the device (or sparse file) in bytes.



Advice

The storage class to which this drive will be assigned must have a PV Size less than or equal to this value.

As of version 9.2, the HPSS disk size limit has increased from 256 TiB to 1 EiB. If you plan to use a disk larger than 256 TiB, contact HPSS support for advice.

If the Starting Offset is nonzero (SAN3P disk only), then the Capacity value cannot be greater than the actual size of the underlying device less the Starting Offset value.

If this value is modified after the disk has been imported into HPSS, it must be emptied, exported, and re-imported.

Starting Offset (SAN3P disk only)

For default disks this field is always zero. For SAN3P disks this is the offset in bytes from the beginning of the disk logical volume at which the Mover will begin using the volume. The space preceding the offset will not be used by HPSS. This value should be a multiple of the **Media Block Size**.



Advice

This field is used only for SAN3p disk devices. The provided default value will specify the offset of the next available byte based on previous devices allocated using the same SAN disk UUID. If changing default value take great care that the new starting offset will not result in the device overlaying any other device using the same SAN disk.

SAN ID (disk only)

The ID for the SAN group. A group is a set of devices that are logically related, with the primary characteristic being that all devices in group are accessible by all clients. Each group is identified with this globally unique group ID. It is assigned to all devices within a SAN group.

Device Flags

The following fields are the device flags used by the Mover.

Read Enabled

An indication of whether the device is available for reading.

Write Enabled

An indication of whether the device is available for writing.

Locate Support (tape only)

An indication of whether the device supports a high speed (absolute) positioning operation.



Advice

This option is supported for 3590, 3590E, 3580, 3592, 9840, 9940, T10000 and GY-8240 devices.

NO-DELAY Support (tape only)

An indication of whether the device supports opening the device with no delay flag set, while allowing tape I/O operation after the open.



Advice

On some tape devices, this will allow for a quicker polling operation when no tape is presently loaded in the device. This field is meaningful for tape devices only.

Removable Media Support (tape only)

An indication of whether the device supports removable media.

TAOS Support (tape only)

An indication of whether the device is set up to use the Spectra Logic TAOS feature. TAOS enables an RAO-like interface for devices which do not support the RAO command set.



Advice

TAOS Support should only be enabled for devices which have already been set up for TAOS with a Spectra Logic library. Discuss your data use cases with Spectra Logic and HPSS before enabling, as TAOS can result in a lower performance in certain scenarios. The TAOS device setup can be confirmed by using device_scan(7) or `lsscsi` on the Mover device. Verify that Spectra Logic supports TAOS on the device type prior to enabling this flag.

Fast Positioning (tape only)

An indication of whether the device should skip section header verification when positioning via LBA.

Disable RAO (tape only)

An indication of whether this device should have RAO (Recommended Access Order) capability disabled. This field has no effect for devices which do not support RAO.

SAN3P-transfer Enabled (disk only)

If ON, SAN3P data transfers to this device will be supported.



Warning: There is a security vulnerability associated with the use of SAN3P. If a user is root on a machine which has access to the SAN (for example, a client machine) then that user has the potential to access or destroy fiber-channel connected disk storage. Two areas of concern: 1) verification that only

authorized users (usually limited to only root or hpss) are granted read and write access to these resource; 2) HPSS administrators should be aware that machines, possibly owned or managed by other groups, which are added to the SAN to facilitate the use of SAN3P transfers will have access to all data on disk and tape resources. If those systems are compromised, or there are individuals authorized for systems privileges on those particular machines, but not necessarily authorized for HPSS administrative access, there is the potential for access and/or damage to HPSS data. These are inherent limitations of SAN implementations that have not yet been addressed by the industry and cannot be remedied by HPSS.

Multiple Mover Tasks (disk only)

If ON, the Mover will allow multiple Mover tasks to access the disk device.

Disable Direct I/O (disk only)

An indication of whether this device should have Direct I/O capability disabled. The HPSS Mover specifies O_DIRECT when working with disks by default in an effort to bypass kernel caching overhead and improve performance. However, for some disk appliances, specifying O_DIRECT can disable appliance-level buffer caching and cause poor performance.

Enable LibAIO (disk only)

An indication of whether this device will utilize the system asynchronous I/O capabilities. The asynchronous I/O capabilities can help increase I/O performance in many circumstances. When enabling LibAIO for a device, the best practices include ensuring that Direct I/O is also enabled. Without Direct I/O enabled, the LibAIO request submission can become blocking (and thus no longer asynchronous) which can limit the amount of parallel I/O achieved and result in poor performance.

Table 27. Recommended settings for tape devices

Device driver	Mover executable	NO-DELAY support	Locate support	TAOS	Fast Positioning	Disable RAO
Linux Native	hpss_mvr_tcp	ON	ON	OFF	ON	OFF

PVR (tape only)

The name of the PVR that handles the removable media operations for the drive.

Drive Address

For tape drives, this is the name or address by which the PVR can access the drive. This field isn't used by the PVL or PVR for disk drives; the administrator may use this as a comment. Many sites duplicate the **Device Name** which makes it valuable as a query of drive information since it will return the device information for the disk case (for example, `lshpss -drv`).



Advice

For StorageTek robots: Drive Address configuration entries correspond to the ACS,Unit,Panel,Drive Number used by ACSLS to identify drives. For example, the first drive in a typical configuration has Drive Address 0,0,10,0.

*For SCSI PVRs: The Drive Address configuration entries correspond to their associated Drive Identifier output from the device_scan(7) tool. The Drive Identifier can be mapped to the device using the **device_scan -t tape** command on the host which will control the devices.*

For IBM robotics, the full identifier string, including any interior spaces, must be present. If Identifier string is:

```
Identifier  = ['T10:IBM|ULT3580-TD3     1210395617']
```

then the Drive Address is

```
T10:IBM|ULT3580-TD3     1210395617
```

*For Spectra Logic robotics, the Drive Address is the output of the **device_scan -i**. However, it must be mapped to the correct drive device as identified by **device_scan -t tape** in order to configure the HPSS device path. These identifiers are close but not identical. For example, **device_scan -i** may indicate:*

```
  1     = [256] ['T10:10110043|4C']
```

*while **device_scan -t** tape may indicate:*

```
Identifier  = ['T10:IBM|ULTRIUM-TD8     101100434C']
```

*These two strings refer to the same device; the library has removed the product and whitespace ("IBM|ULTRIUM-TD8   ") that is displayed in the **device_scan -t** output. However, the string from **device_scan -i** is what is required, since that is what the SCSI PVR will use to detect the device. The Drive Address should be:*

```
T10:10110043|4C
```

For operator mounted drives: For manually mounted drives, the drive address string is displayed to operators when an explicit drive is selected for mounting. The drive address should therefore be a string that easily communicates to an operator the drive in question (that is, a name matching the label on the exterior of the drive).

Controller ID

An identifier that signifies that a set of disk or tape devices have shared bandwidth (e.g. are in the same storage unit, adapter, etc).



Advice

This field is used to attempt to mount individual volumes of a set of striped media on individual controllers, when possible, to prevent contention on the data path. It is recommended that the drives that are on the same adapter/bus have the same Controller ID.

For disk volumes, this field is used when the environment variable HPSS_SS_SCHEDULE_BY_CONTROLLER is set in `env.conf` *to balance the usage of disk volumes by the controller ID rather than by Mover ID.*

Polling Interval (tape only)

The number of seconds to wait between polling requests performed by the PVL to determine if any media is present in the drive. This field is disabled by default. Use "-1" to disable polling. Values of "0" to "14" are not valid. If the drive is an operator-mounted tape drive, then this field shouldn't be disabled.



Advice

It is recommended that this field be set to a value greater than 14 seconds for operator-mounted tape drives and disabled for library managed tape drives. In normal operations for library managed tape drives (for example, STK SL8500), drive polling should not be required. In reality there are occasions where polling may detect a non-HPSS cartridge or an unaccounted for HPSS cartridge. However, the use of the drive polling (originally in place for Operator PVRs) is associated with high overhead (one thread per drive that is polling). A less expensive alternative that addresses this condition is currently being pursued. In the meantime, if you're concerned with monitoring for unexpected cartridges in library managed tape drives, it is recommended that you set the polling interval to a value no less than "900" (15 minutes).

Maintenance Date

The time and date when the drive last had maintenance performed on it. This date must be entered manually and is provided for administrative information purposes only.

Mounted Volume (disk / object only)

The eight-character name of the volume mounted on the disk drive or object store volume.

Comment

This field provides a 128-character buffer in the PVL drive metadata which gives the administrator the opportunity to associate miscellaneous text with a device/drive. For example, a site may want to place a comment in this field that the drive is out of service or being used by another system.

Library Unit ID (tape only)

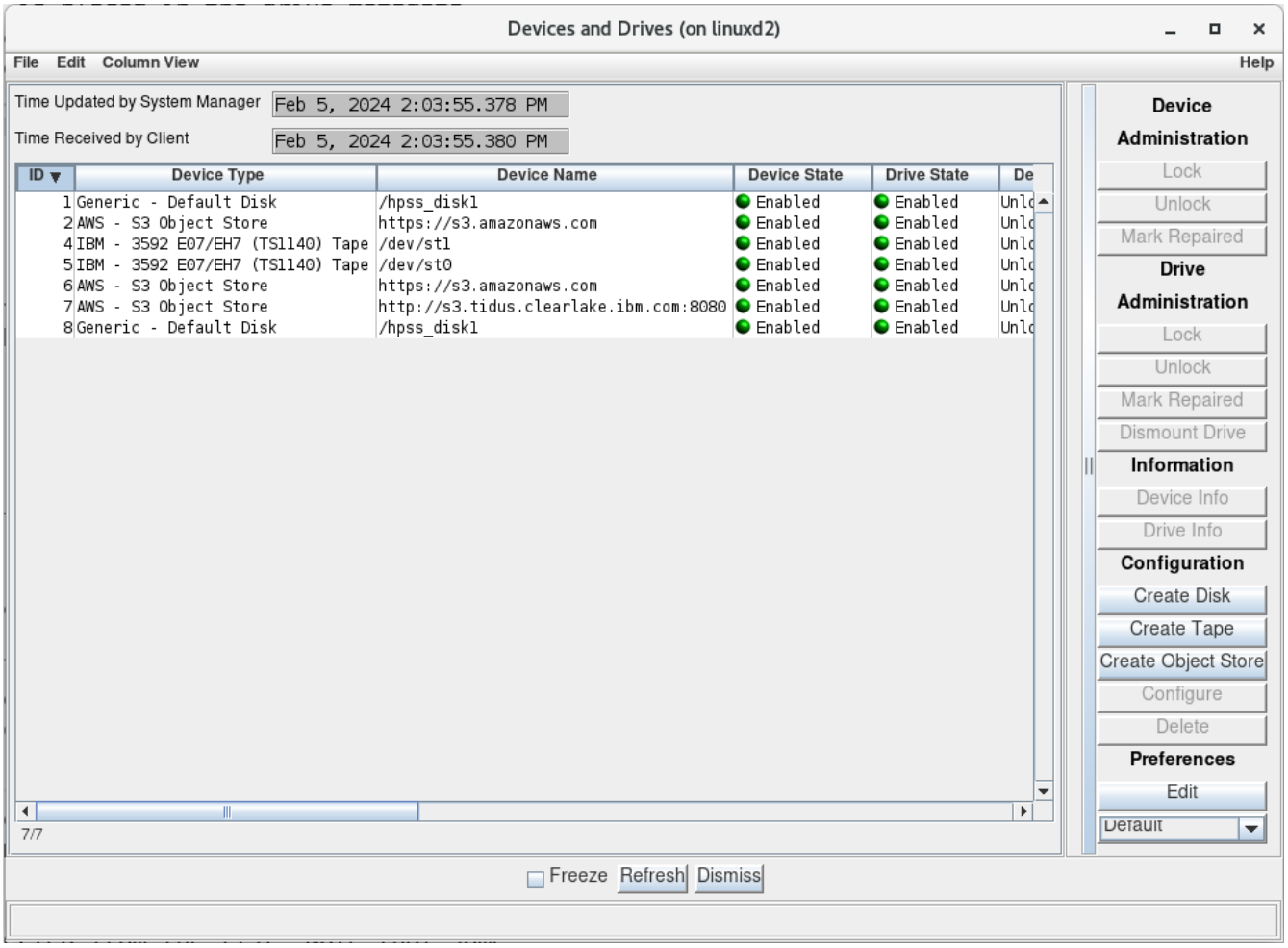
Since library architecture varies, the method to associate related drives at a unit level will be determined through the use of a "unit id" that is placed in the drive metadata. This will allow the administrator to analyze their specific library configuration and apply their strategy as they feel fit. The use of "unit id" allows the library specifics to be abstracted from the PVL. Library details are the domain of the PVR. See **Defer Dismount Exempt Count** (PVR Server Configuration).



Advice

Devices configured for AWS Tape Gateway must have SCSI reservations disabled. This can be done by adding "HPSS_MVR_DISABLE_RESERVATIONS=<DeviceId>" in /var/hpss/etc/env.conf on the Mover node.

18.1.1. Devices and Drives window



This window allows you to view the list of configured Mover devices and PVL drives. It also provides a number of function buttons, which allow certain operations to be performed on devices or drives.

The *Device and Drive List Preferences* window may be used to select the device/drives which will be displayed. Select the columns to be displayed in the list from the Column View menu.

Most of the function buttons to the right of this list require that one or more device/drive entries be selected from the list. Note that some of the fields in the table may not be visible, depending on the Column View menu settings.

Field descriptions

Device and Drive List

ID

The unique device/drive ID number defined when the device/drive is configured.

Device Type

The type of the device.

Device Name

The name used by the Mover to access the device, usually the pathname of a UNIX device special file (such as `/dev/rmt0`).

Device State

The current operational state of the device, as reported by its controlling Mover. The possible states are:

- `Enabled` indicates the device is working normally.
- `Suspect` indicates the Mover has detected errors on the device, but it is still functioning.
- `Disabled` indicates the device is locked, which makes it unavailable for use by HPSS.
- `Unknown` indicates the state of the device is not known to SSM; this is usually caused by the controlling Mover being down or disconnected from SSM.

Device Admin State

The current administrative state of the device, as reported by its controlling Mover. The possible states are:

- `Locked` indicates the device is locked and unavailable for use by HPSS.
- `Unlocked` indicates the device is unlocked and available for use by HPSS.
- `Unknown` indicates the state of the device is not known to SSM; this is usually caused by the controlling Mover being down or disconnected from SSM.

Mover

The descriptive name of the Mover that controls the device.

Mover Host

The hostname of the Mover node (the node where the Mover runs).

Drive Address

The name used by the PVL/PVR to access the drive.

Drive Flag

This field describes possible actions that the PVL needs to perform for the drive. Possible flags are:

- `Clear` indicates nothing needs to be done.
- `Modified` is set by SSM when the drive has been modified. This value is reserved for use in a planned feature. It is not used at this time.
- `PVR/Mover Notify Pending` indicates the PVL needs to notify the associated PVR and Mover that the drive has been created or deleted.
- `PVR Notify Pending` indicates the PVL needs to notify the associated PVR that the drive has been created or deleted.

- `Mover Notify Pending` indicates the PVL needs to notify the associated Mover that the drive has been created or deleted.
- `Abort PVR/Mover Notify` indicates the PVL is aborting a pending notification.

Drive State

The current operational state of the drive, as reported by the PVL. The possible states are:

- `Enabled` indicates the drive is working normally.
- `Disabled` indicates the drive is not available for use by HPSS. A drive may be disabled if it is locked by an SSM user, if an error occurred when mounting or dismounting a cartridge on the drive, or if the Core Server was unable to connect to the Mover after a tape was mounted on the drive.
- `Broken` indicates the PVL itself has detected a fatal error and is shutting itself down.
- `Unknown` indicates the state of the device is not known to SSM; this is usually caused by the PVL being down or disconnected from SSM.

Drive Admin State

The current administrative state of the drive, as reported by the PVL. The possible states are:

- `Locked`, indicating the drive is locked and unavailable for use by HPSS.
- `Unlocked`, indicating the drive is unlocked and available for use by HPSS.
- `Unknown`, indicating the state of the drive is not known to SSM; this is usually caused by the PVL being down or disconnected from SSM.

Library Unit ID (tape only)

Since library architecture varies, the method to associate related drives at a unit level will be determined through the use of a "unit id" that is placed in the drive metadata. This will allow the administrator to analyze their specific library configuration and apply their strategy as they feel fit. The use of "unit id" allows the library specifics to be abstracted from the PVL. Library details are the domain of the PVR. See **Defer Dismount Exempt Count** (PVR Server Configuration).

Comment

This field provides 128 characters of storage in the PVL drive metadata in which the administrator may store text messages associated with a device/drive. For example, a site may want to place a comment in this field that the drive is out of service or being used by another system.

PVR

The descriptive name of the PVR that handles removable media operations for the drive. This will be blank if the device/drive does not support removable media.

Mounted Volume

The name of the volume that is currently mounted as reported by the PVL.

Bytes Read

Number of bytes read from the device/drive as reported by the Mover.

Bytes Written

Number of bytes written to the device/drive as reported by the Mover.

State

The overall operational state of the device/drive. If this is not `Enabled`, check the individual **Device State** and **Drive State** fields to help determine why.

Device Administration buttons

This group of buttons affects selected Mover devices. All the buttons are disabled unless one or more devices are selected (see figure above).

Lock

Locking the selected devices serves no functional purpose. When the device is locked, the information is not transferred to the PVL and thus the PVL will continue to schedule the associated PVL drive (see **Lock** button in the **Drive Administration Buttons** below). It is also not transferred to the Core Server which will continue to read, write, create, or mount the volume (see **VV Condition** in the *Core Server Volume Disk/Tape* window). This leads to many red Alarms. After clicking the Device Administration **Lock** button, you are prompted to confirm your request before it is actually executed. The window's message line reports when locking begins. When the request is complete, another message tells how many requests were successful.

Unlock

Unlock the selected devices, making them available to the Mover. You are prompted to confirm your request before it is actually executed. The window's message line reports when unlocking begins. When the request is complete, another message tells how many requests were successful.

Mark Repaired

Marks the selected devices as "repaired". Sometimes the device states (such as operational state) will continue to indicate an error condition after the cause of the error has been fixed. If so, you can mark a device repaired to instruct the Mover to clear its error states. Note that this does nothing, either in hardware or software, to actually repair an error condition. Also, if you mark a device repaired when it still has a problem, the error states will be cleared but may quickly return to their previous values.

You are prompted to confirm your request before it is actually executed. The window's status bar displays a message when marking begins, and when the request is complete, another message tells how many requests were successful.

Drive Administration buttons

This group of buttons affects selected PVL drives. All the buttons are disabled unless one or more drives are selected (see figure above).

Lock

Lock the selected drives, making them unavailable to HPSS. When a drive is locked, the PVL will no longer schedule the PVL drive. When locking a tape drive due to a cartridge problem (such as for a stuck tape), we recommend canceling the PVL Job associated with the cartridge. We also

recommend changing the Condition of the tape volume in the Core Server that owns it so that no attempts will be made to perform I/O operations on the volume. The "Down" condition is provided for this purpose. Locking the PVL drive does not automatically change the volume's Condition. After clicking the Drive Administration **Lock** button, you are prompted to confirm your request before it is performed. The window's message line reports when locking begins. When the request is complete, another message tells how many requests were successful.

Unlock

Unlock the selected drives, making them available to HPSS. You are prompted to confirm your request before it is actually performed. The window's message line reports when unlocking begins. When the request is complete, another message tells how many requests were successful.

Mark Repaired

Notify the PVL and Mover to clear any error states for the selected drives. Sometimes the drive states (such as operational state) will continue to indicate an error condition after the cause of the error has been corrected. If so, you can mark a drive repaired to instruct the Mover or PVL to clear its error states. Note that this does nothing, either in hardware or software, to actually repair an error condition. Also, if you mark a drive repaired when it still has a problem, the error states will be cleared but may quickly return to their previous values.

You are prompted to confirm your request before it is actually executed. The window's status bar displays a message when marking begins, and when the request is complete, another message tells how many requests were successful.

Dismount

Dismount the selected drives. You are prompted to confirm your request before it is actually executed. The window's status bar displays a message when dismounting begins. When the request is complete, another status message tells how many requests were successful.

Information buttons

This group of buttons retrieves the device-managed or drive-managed object for one or more selected list items. All the buttons are disabled unless one or more devices/drives are selected.

Device Info

Open the *Mover Device Information* window for the selected item. If the selected item's Mover cannot be contacted, this button will be desensitized. While this window is open, it will continue to update reflecting the latest device data from the Mover.

Drive Info

Open the *PVL Drive Information* window for the selected item. If the PVL cannot be contacted, this button will be desensitized. While this window is open, it will continue to update reflecting the latest drive data from the PVL.

Configuration buttons

This group of buttons allows you to do device/drive configuration.

Create Disk

This button is always active. Clicking on it opens the *Disk Device Configuration* window in Add mode, allowing you to create a new disk device and drive.

Create Tape

This button is always active. Clicking on it opens the *Tape Device Configuration* window in Add mode, allowing you to create a new tape device and drive.

Create Object Store

This button is always active. Clicking on it opens the *Object Store Device Configuration* window in Add mode, allowing you to create a new object store device and drive.

Configure

This button is active if one or more devices/drives are selected. Clicking the button opens the *Disk/Tape/Object Store Device Configuration* window for the selected devices and drives, allowing you to view, delete, clone, or update the configurations.

Delete

This button is active if one or more device/drives is selected from the list. A dialog will be displayed for each device/drive selected asking the user to confirm that the device/drive configuration is to be deleted.

Preferences

See *Devices and Drives* window for information on changing preferences for the *Devices and Drives* window.

18.1.2. Enable variable block sizes for tape devices

All tape devices that will be used for HPSS data must be set to handle variable block sizes (to allow for the ANSI standard 80-byte volume label and file section headers). This requirement is described in detail in the Tape devices section. Refer to that section to be certain your tape devices are configured correctly.

18.1.3. Changing a drive's configuration

Devices and drives can be updated dynamically (that is, while the PVL, associated Movers, and associated PVRs (for tape) are up). Many of the same required conditions for drive deletion must be met. The drive must be locked without any activity (drive state = FREE). Additionally for disk, the volume must be unallocated. However, there are some drive attributes for which the configuration metadata can be updated dynamically via the *PVL Drive Information* window and there are some device attributes for which the configuration metadata is not updated dynamically but the server "in memory" values can be changed dynamically (but not preserved when the Mover is restarted) via the *Mover Device Information* window without the drive locking/free state step.

The device and drive configuration entry can be updated using the *Disk Device Configuration*, *Tape Device Configuration* window, or *Object Store Configuration*. The *Disk Device Configuration*, *Tape Device Configuration*, and *Object Store Device Configuration* windows are updated dynamically.

These windows are opened by clicking the **Configure** button on the *Devices and Drives* window. Changing any settable field on the disk/tape/object store device configuration windows requires all servers involved in the update to be running:

- the Mover that is currently managing the device
- the new Mover that will manage the device (if changing)
- the PVR that is currently managing cartridges for this drive (if tape)
- the PVR will manage cartridges for this drive (if tape and if changing)

An option exists which allows the PVR to make a configuration update without the associated servers running. This is intended for an abnormal condition and should be applied sparingly and with caution. If the PVL cannot communicate with a Mover, the PVL communicates the condition to the SSM. The SSM will prompt the administrator with an override option which if set will direct the PVL to update metadata without successfully communicating to the associated Mover. It's expected that this option may be used in a failover scenario where a Mover platform has crashed and drives are moved to another platform while the failed platform is repaired or replaced. The HPSS drive move would be preceded by drive hardware changes to make the drives available at the moved platform.

Some PVL drive configuration attributes can be updated dynamically using the *PVL Drive Information* window.

The settable fields in this window are updated dynamically (that is, saved to metadata and used by the PVL upon successful update).

It is also possible to make temporary updates to some Mover device attributes parameters by changing the settable fields in the *Mover Device Information* window instead of the configuration window. These changes stay in effect until the Mover is recycled.

18.1.4. Deleting a drive's configuration

Devices and drives are deleted dynamically. This means that they must be deleted while the system is running. Specifically the PVL must be up; additionally it is expected that the associated Mover and associated PVR (for tape) are also up since the PVL needs to notify these servers about the deletion. If the PVL is unable to successfully notify the associated Mover and/or PVR about the deleted device/drive, it will retry several times; each failure will be logged as an ALARM. After the last failed attempt to notify the associated Mover and/or PVR, the PVL will abandon retrying; the administrator is advised to bounce the abandoned associated Mover (if it was up during the failed retries) and/or to reinitialize the abandoned associated PVR (if it was up during the failed retries) to ensure that the device/drive is completely deleted and thus can no longer be used.

Likewise, a previously added device/drive cannot be deleted until the PVL has successfully notified the associated Mover and PVR (for tape) about the previous add. If the administrator attempts to delete a device/drive with a previously added Drive ID that is still pending add notification, the administrator will get a BUSY failure and the pending notification will be aborted. An ALARM will be logged informing the administrator that the delete failed since the Mover/PVR add/create notification is pending. Another ALARM will be logged informing the administrator that the previous drive create notification will be aborted and to try later; wait a minute or so before

retrying in order to give the PVL time to abort the notification. The **Drive Flag** field in the *Devices and Drives* window and *PVL Drive Information* window will report whether or not a particular drive is pending Mover/PVR notification; of course, this is only viewable while the drive exists.

There are a number of situations in which the PVL won't allow the device/drive to be deleted:

- If the device/drive is still attempting to notify the Mover/PVR about being added
- If the device/drive is in the process of aborting Mover/PVR notification
- If the drive is in use by the PVL
- If the drive is not locked
- For disk: If storage resources haven't been deleted from the disk device/drive
- For disk: If the physical volume hasn't been exported from the disk device/drive

If it is only necessary to take a drive out of service for a finite period of time, the administrator should follow the instructions in the *Devices and Drives* window section to lock the drive and prevent its use rather than permanently removing the drive as described in this section.

The Device and Drive configuration entry can be deleted using the *Devices and Drives* list window (see the *Devices and Drives* window section).

Select the correct device/drive configuration and click on the **Delete** button to delete the configuration. Once the drive configuration is deleted, HPSS will no longer be able to use the drive's resources.

18.2. Monitoring devices and drives

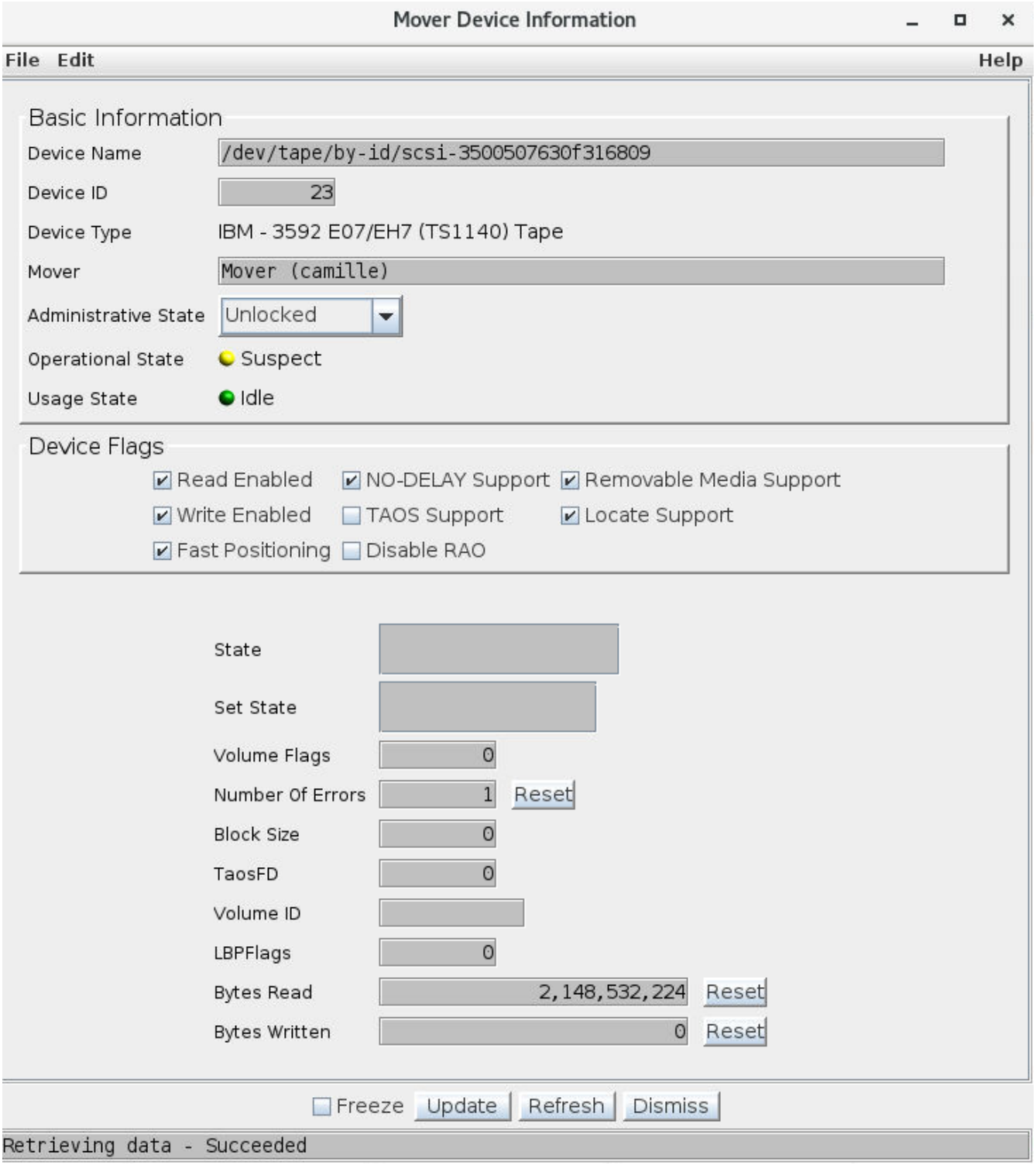
After the devices/drives are configured and are made available for service, the device/drive status may be monitored via several SSM windows.

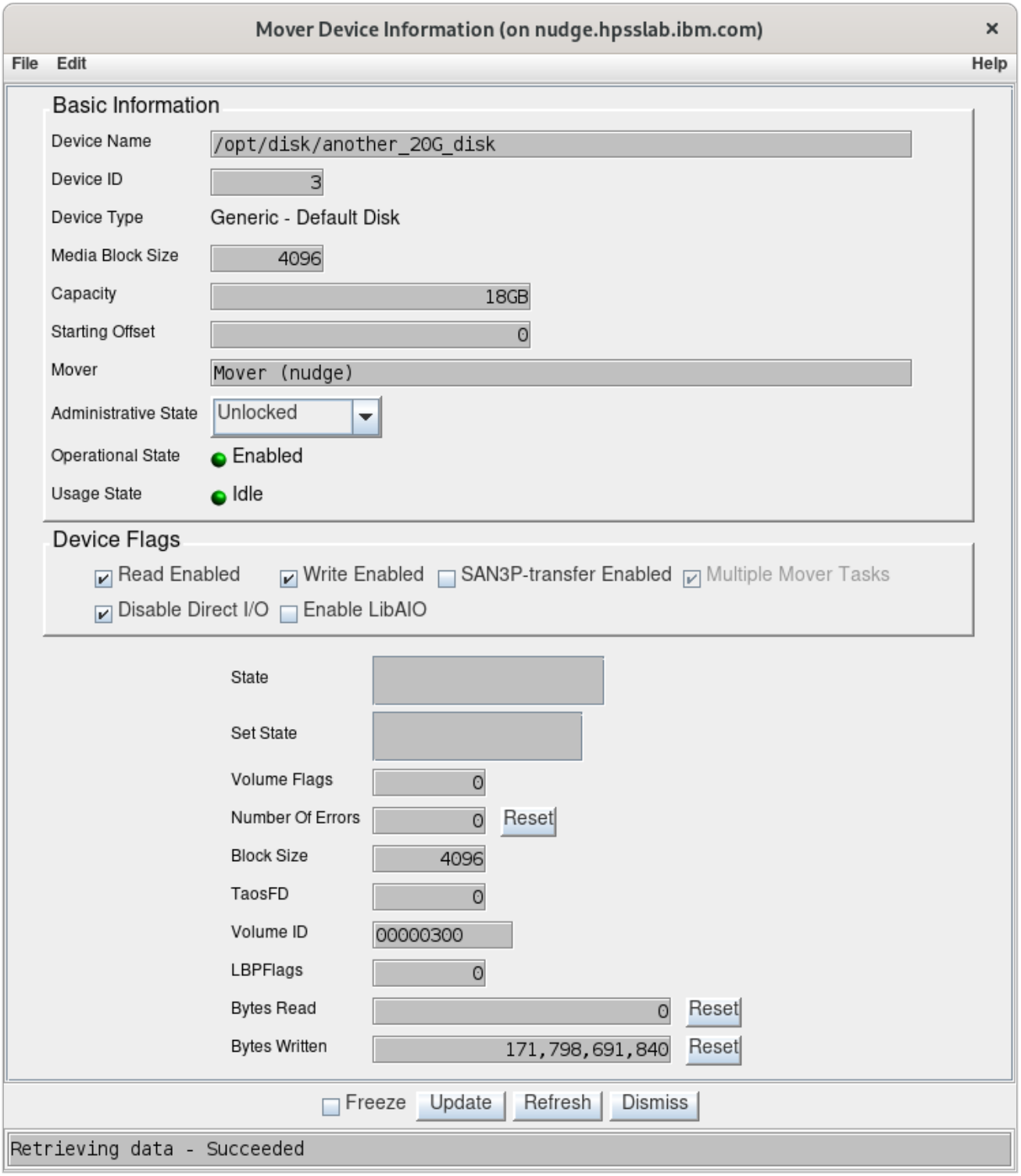
When a device/drive experiences a problem, the PVL and/or the Mover issues one or more alarms to inform SSM users of the problem and to notify SSM of the changes in the device/drive state. SSM reports the state change on the **Devices and Drives** status field on the *HPSS Health and Status* window. SSM also reports the new state on the *Devices and Drives* list window, the *Mover Device Information* window, and the *PVL Drive Information* window if they are being displayed at the time.

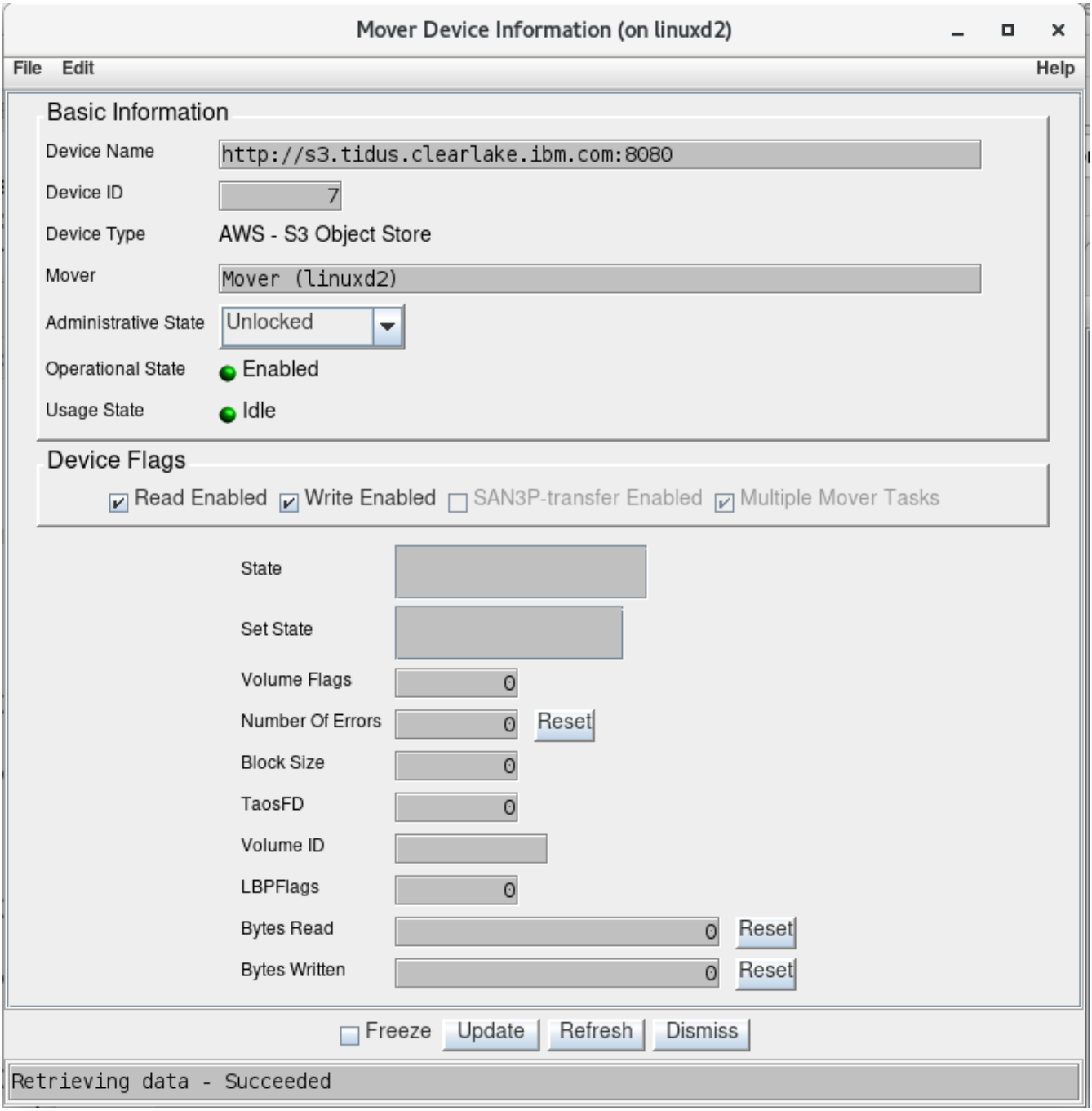
If an HPSS device or drive needs to be locked or unlocked or if any device/drive configuration needs to be done, the *Devices and Drives* list window is the first stop (for detailed information, see the *Devices and Drives* window section). It facilitates drive locking and unlocking, and it also lets administrators access the following windows:

- *Disk/Tape/Object Store Device Configuration* is used to configure, modify, or delete device and drive configurations. See Configure a new device and drive for more information.
- *Mover Device Information* is used to view or update the state of the Mover device. See *Mover Device Information* window for more information.
- *PVL Drive Information* is used to view or update the state or configuration of the PVL drive. See *PVL Drive Information* window for more information.

18.2.1. Mover Device Information window




Mover Device Information (on nudge.hpsslab.ibm.com)
File
Edit
Help
Basic Information
Device Name
/opt/disk/another_20G_disk
Device ID
3
Device Type
Generic - Default Disk
Media Block Size
4096
Capacity
18GB
Starting Offset
0
Mover
Mover (nudge)
Administrative State
Unlocked
Operational State
Enabled
Usage State
Idle
Device Flags
Read Enabled
Write Enabled
SAN3P-transfer Enabled
Multiple Mover Tasks
Disable Direct I/O
Enable LibAIO
State
Set State
Volume Flags
0
Number Of Errors
0
Reset
Block Size
4096
TaosFD
0
Volume ID
00000300
LBPFlags
0
Bytes Read
0
Reset
Bytes Written
171,798,691,840
Reset
Freeze
Update
Refresh
Dismiss
Retrieving data - Succeeded



The *Mover Device Information* window reports the current statistics for the device, such as the workload history of the device since the startup of the controlling Mover. The *Mover Device Information* window can also be used to lock and unlock a Mover device. Additionally, it can be used to control the I/O aspects of the device.

Note: locking the Mover device generally is not helpful; see *Devices and Drives* window.

Changes to the **Administrative State** cause a confirmation window to pop up. If the user confirms the request, modifications made to all fields on the window will be sent to the Mover and become effective immediately.

Field descriptions

Device Name

The name by which the Mover accesses the device; usually the pathname of a UNIX device

special file (such as `/dev/rmt0`).

Device ID

The ID number unique to this device. The same ID number also refers to the corresponding PVL drive.

Device Type

The HPSS media type which describes the physical device.

Media Block Size (disk only)

The block size for the device.

Capacity (disk only)

The size of the disk device in bytes.

Starting Offset (SAN3P disk only)

The offset in bytes from the beginning of the disk logical volume at which the Mover will begin using the volume. The space preceding the offset will not be used by HPSS.

Mover

The descriptive name of the Mover which controls the device.

Administrative State

This field allows you to modify the state of the device. Click on the option menu button to pop up a list of valid states. These are:

- `Locked` makes the device unavailable for HPSS requests.
- `Unlocked` makes the device available for HPSS requests.
- `Mark Repaired` tells the Mover to clear any error status for the device. This can be useful if you think a problem has been fixed, but the Mover is unaware of it. This does not do anything, in hardware or software, to fix a problem; it only clears error indicators.

Operational State

Indicates the device's ability to handle HPSS requests. Possible values for this field are:

- `Enabled`, indicating the device is available for HPSS requests.
- `Disabled`, indicating the device is unavailable for HPSS requests, possibly caused by setting the "Locked" Administrative State.

Usage State

Indicates the state of the Mover's control over the device. Possible values for this field are:

- `Active` indicates a Mover task currently has control of the device.
- `Idle` indicates no Mover task is controlling the device.

Device Flags

This group of buttons defines various device characteristics. When a button is ON, it means that the corresponding flag is set.

Read Enabled

An indication of whether the device is available for reading.

Write Enabled

An indication of whether the device is available for writing.

Locate Support (tape only)

An indication of whether the device supports a high speed (absolute) positioning operation.



Advice

This option is supported for IBM 3590, 3590E, 3580, 3592, StorageTek 9840, 9940, T10000 and GY-8240 devices.

NO-DELAY Support (tape only)

An indication of whether the device supports opening the device with no delay flag set, while allowing tape I/O operation after the open.



Advice

On some tape devices, this will allow for a quicker polling operation when no tape is presently loaded in the device. This field is meaningful for tape devices only.

TAOS Support (tape only)

An indication of whether the device is set up to use the Spectra Logic TAOS feature. TAOS enables an RAO-like interface for devices which do not support the RAO command set.



Advice

TAOS Support should only be enabled for devices which have already been set up for TAOS with a Spectra Logic library. Discuss your data use cases with Spectra Logic and HPSS before enabling, as TAOS can result in a lower performance in certain scenarios. The TAOS device setup can be confirmed by using device_scan(7) or `lsscsi` *on the Mover device. Verify that Spectra Logic supports TAOS on the device type prior to enabling this flag.*

Fast Positioning (tape only)

An indication of whether the device should skip section header verification when positioning via LBA.

Disable RAO (tape only)

An indication of whether this device should have RAO (Recommended Access Order) capability disabled. This field has no effect for devices which do not support RAO.

Removable Media Support (tape only)

An indication of whether the device supports removable media.

SAN3P-transfer Enabled (disk only)

If ON, SAN3P data transfers to this device will be supported.



Warning *- There is a security vulnerability associated with the use of SAN3P. If a user is root on a machine which has access to the SAN (for example, a client machine) then that user has the potential to access or destroy fiber-channel connected disk storage. Two areas of concern: 1) verification that only authorized users (usually limited to only root or hpss) are granted read and write access to these resource; 2) HPSS administrators should be aware that machines, possibly owned or managed by other groups, which are added to the SAN to facilitate the use of SAN3P transfers will have access to all data on disk and tape resources. If those systems are compromised, or there are individuals authorized for systems privileges on those particular machines, but not necessarily authorized for HPSS administrative access, there is the potential for access and/or damage to HPSS data. These are inherent limitations of SAN implementations that have not yet been addressed by the industry and cannot be remedied by HPSS.*

Multiple Mover Tasks (disk only)

If ON, the Mover will allow multiple Mover tasks to access the disk device.

Disable Direct I/O (disk only)

An indication of whether this device should have Direct I/O capability disabled. The HPSS Mover specifies O_DIRECT when working with disks by default in an effort to bypass kernel caching overhead and improve performance. However, for some disk appliances, specifying O_DIRECT can disable appliance-level buffer caching and cause poor performance.

Enable LibAIO (disk only)

An indication of whether this device will utilize the system asynchronous I/O capabilities. The asynchronous I/O capabilities can help increase I/O performance in many circumstances. When enabling LibAIO for a device, the best practices include ensuring that Direct I/O is also enabled. Without Direct I/O enabled, the LibAIO request submission can become blocking (and thus no longer asynchronous) which can limit the amount of parallel I/O achieved and result in poor performance.

State

A set of flags describing the current state of the device. The flag states are presented as a list of flag descriptions for each flag which is currently ON:

- `Mounted` indicates a media volume is mounted in the device.
- `Open for Read` indicates the device is open for reading.
- `Open for Write` indicates the device is open for writing.
- `Busy` indicates the device is in use.
- `Error Encountered` indicates an error was encountered on the device.
- `Hit End-Of-Tape` indicates the end of a tape was detected.

- `Client Off Midblock` indicates the device position is in the middle of a block.
- `Data to be Synced` indicates there are data that need to be flushed.
- `Last Op Read` indicates the last operation on the drive was a read.
- `Logical Block Support` indicates the device supports Logical Block Addressing.

Set State

A set of flags showing device state changes which are pending. The flag states are presented as a list of flag descriptions for each flag which is currently ON:

- `Toggle Read Mode` - Toggle the current read mode value on the device.
- `Toggle Write Mode` - Toggle the current write mode value on the device.
- `Toggle In-Use Mode` - Release the device from the in use state.
- `Reset Error Count` - Reset the device **Number of Errors** field
- `Reset Bytes Read` - Reset the **Bytes Read** field.
- `Reset Bytes Written` - Reset the **Bytes Written** field.
- `Set to Locked` - Set the device **Administrative State** to locked.
- `Set to Unlocked` - Set the device **Administrative State** to unlocked.
- `Set to Repaired` - Clear any error status for the device.

Volume Flags

Flags which further describe that describe the volume type.

Number of Errors

The number of errors that have occurred on this device since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the count to zero.

Block Size

The size, in bytes, of each data block on this device.

Volume ID

The label (name) of the volume which is currently mounted on this device. If no volume is currently mounted, this field will be blank.

Bytes Read

The number of bytes that have been read from this device since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the count to zero.

Bytes Written

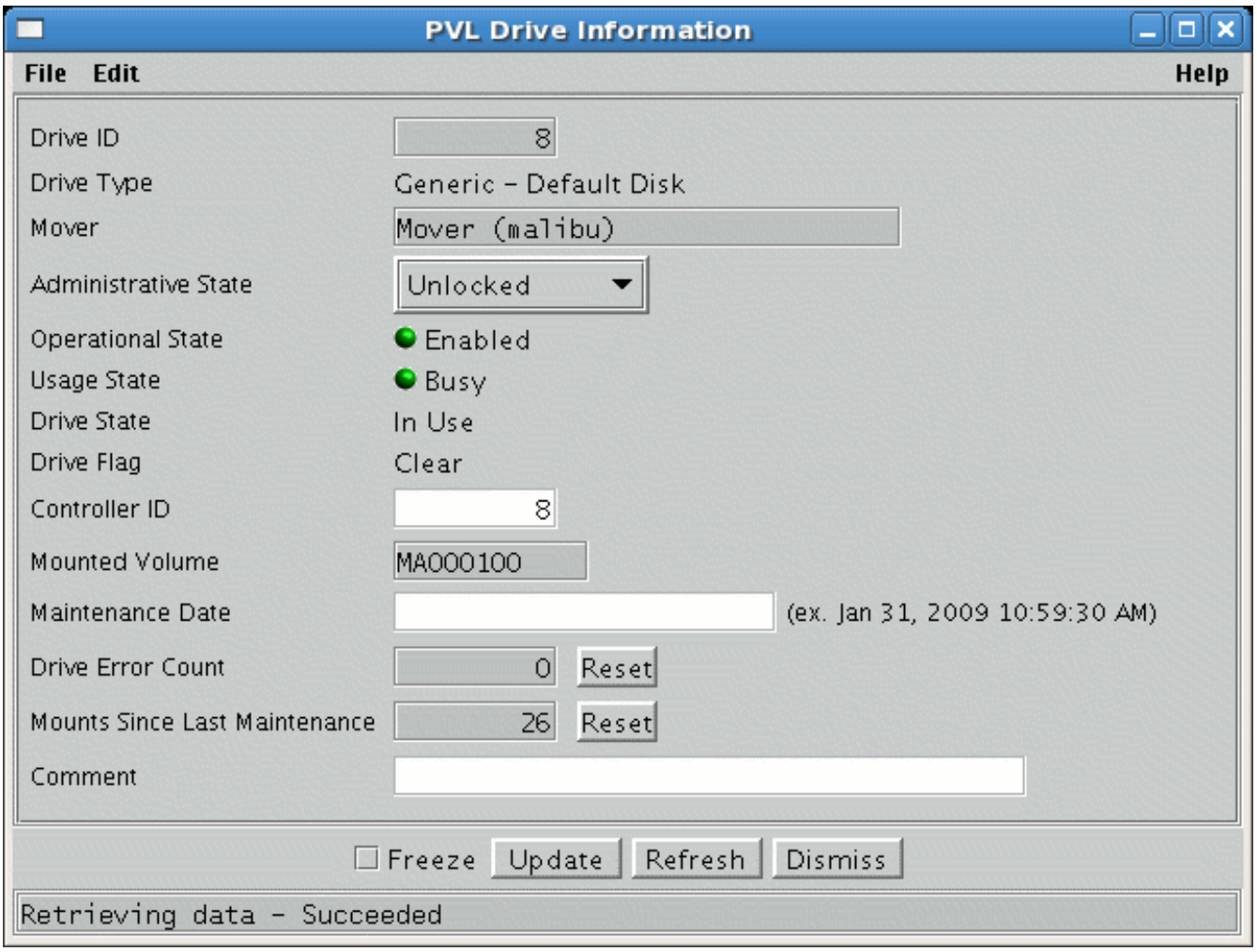
The number of bytes that have been written to this device since the last time this field was cleared. Click on the **Reset** button to the right of the field to reset the count to zero.

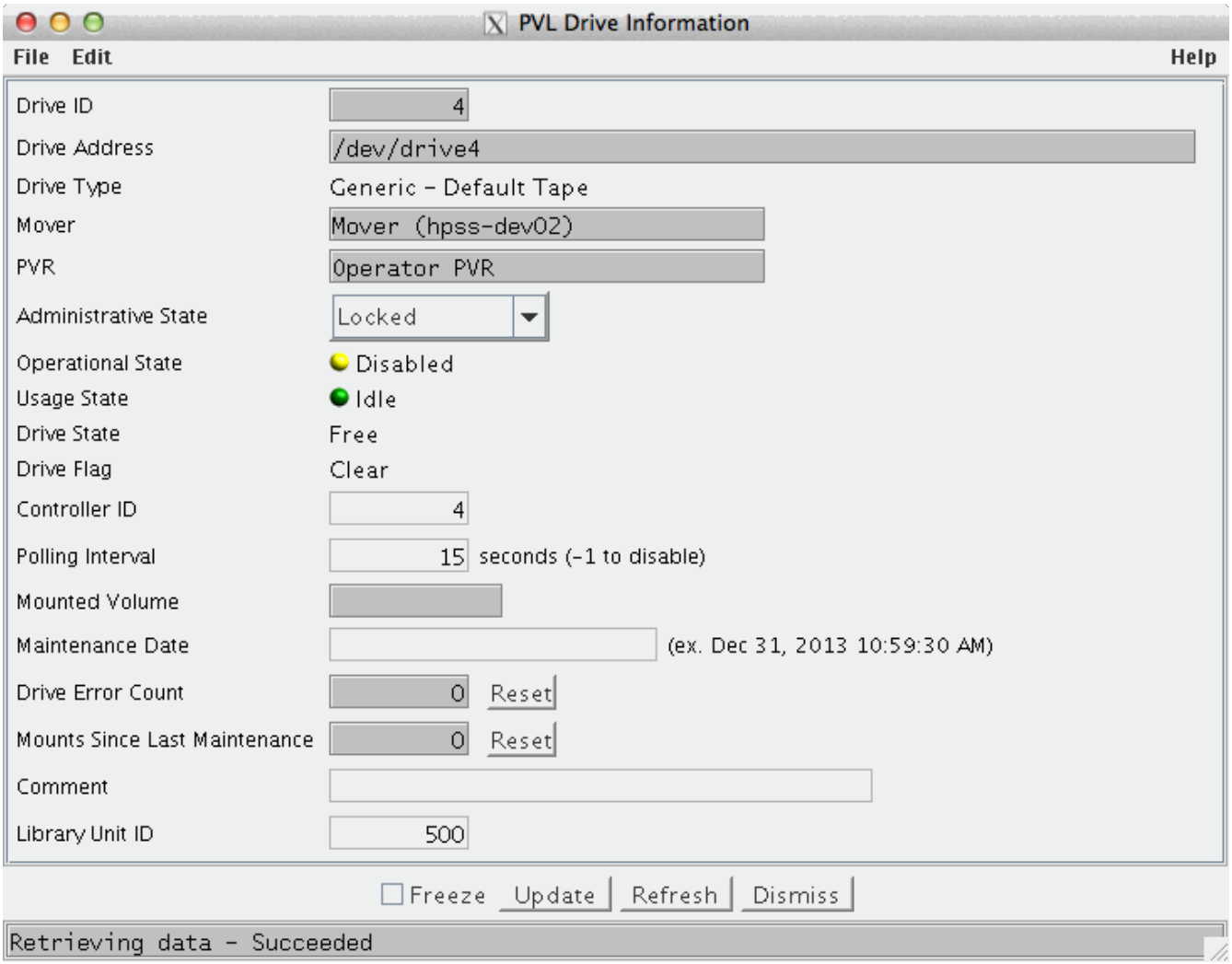
Operational notes

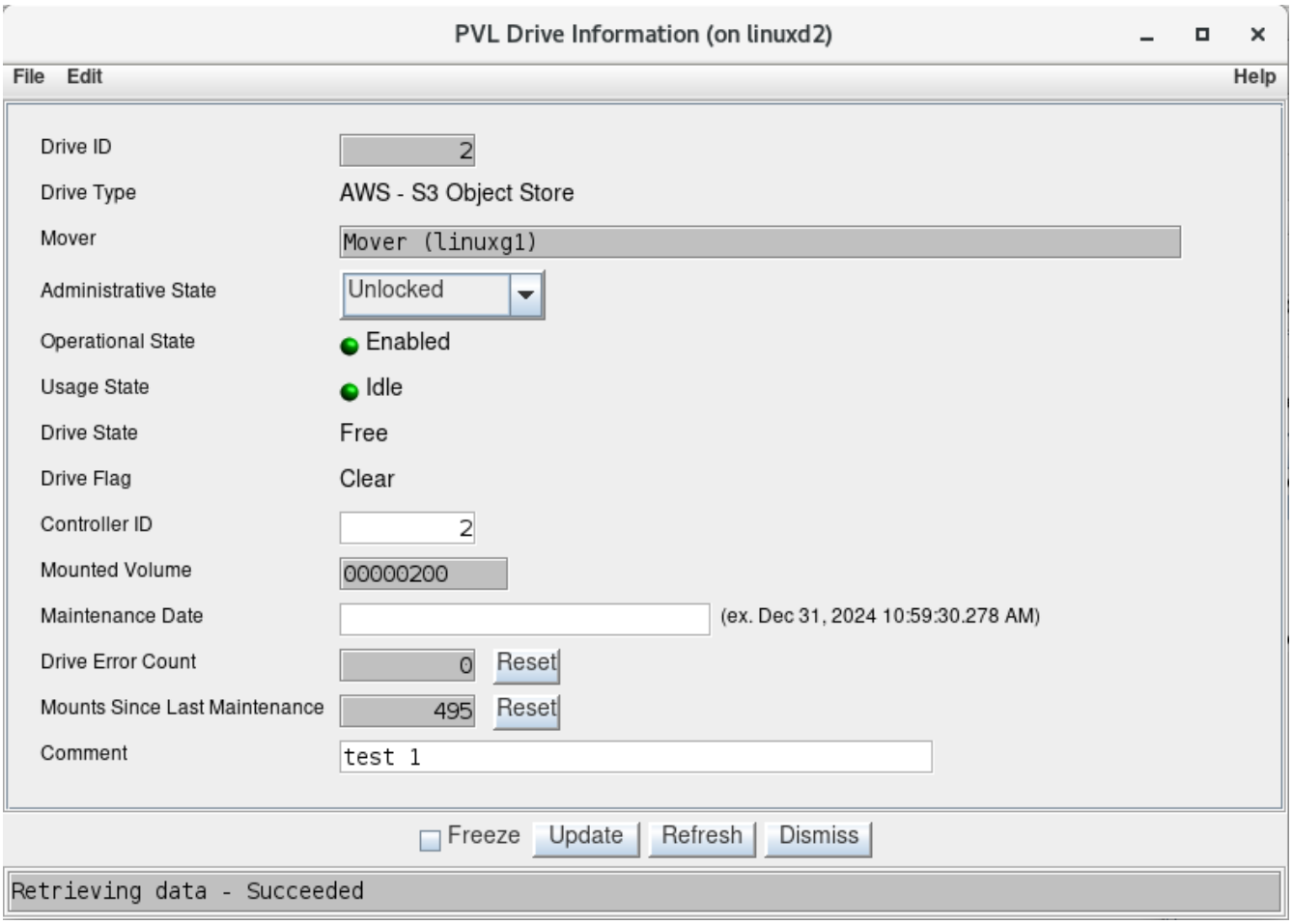
- Setting device attributes through this window will not cause the configuration metadata to be updated.

- The device statistics fields (number of bytes read, number of bytes written, number of errors) are all initialized to zero when the Mover is initialized.
- When modifying the **Administrative State**, toggling the device flags or resetting the device statistics, changes do not take place until the device is no longer in use. These modifications will be pending until the device is released and no longer busy.

18.2.2. PVL Drive Information window




PVL Drive Information
File Edit Help
Drive ID 4
Drive Address /dev/drive4
Drive Type Generic - Default Tape
Mover Mover (hpss-dev02)
PVR Operator PVR
Administrative State Locked
Operational State Disabled
Usage State Idle
Drive State Free
Drive Flag Clear
Controller ID 4
Polling Interval 15 seconds (-1 to disable)
Mounted Volume
Maintenance Date (ex. Dec 31, 2013 10:59:30 AM)
Drive Error Count 0 Reset
Mounts Since Last Maintenance 0 Reset
Comment
Library Unit ID 500
Freeze Update Refresh Dismiss
Retrieving data - Succeeded


PVL Drive Information (on linuxd2)
File Edit Help
Drive ID 2
Drive Type AWS - S3 Object Store
Mover Mover (linuxg1)
Administrative State Unlocked
Operational State Enabled
Usage State Idle
Drive State Free
Drive Flag Clear
Controller ID 2
Mounted Volume 00000200
Maintenance Date (ex. Dec 31, 2024 10:59:30.278 AM)
Drive Error Count 0 Reset
Mounts Since Last Maintenance 495 Reset
Comment test 1
Freeze Update Refresh Dismiss
Retrieving data - Succeeded

This window allows you to view and update the information associated with an HPSS drive. The *PVL Drive Information* window is typically used to lock and unlock drives since newly configured drives are locked by default and must be unlocked to be used. It may also be used to determine which volume is mounted on the drive when the drive reports a mount error condition.

Any changes made to fields on this window are saved in metadata and sent directly to the PVL and thus are effective immediately. Thus the changes are dynamic.

Field descriptions

Drive ID

The ID number unique to this drive. The same ID number also refers to the corresponding Mover device.

Drive Address (tape only)

The name/address by which the drive's PVR can access it. This information is used in communications between the PVL and the PVR that controls removable media operations for the drive.

Drive Type

The HPSS drive type which describes the physical drive.

Mover

The descriptive name of the Mover which moves data to and from this drive.

PVR (tape only)

The descriptive name of the PVR used to control this drive. This field is only meaningful for tape drives.

Administrative State

This field allows you to modify the state of the drive. The options are:

- `Locked` makes the drive unavailable for HPSS requests.
- `Unlocked` makes the drive available for HPSS requests.
- `Mark Repaired` tells the PVL to clear any error status for the drive. This can be useful if you think a problem has been fixed, but the PVL is unaware of it. This does not do anything, in hardware or software, to fix a problem; it only clears error indicators.

Operational State

Indicates the drive's ability to handle HPSS requests. Possible values for this field are:

- `Enabled` indicates the drive is available for HPSS requests.
- `Disabled` indicates the drive is unavailable for HPSS requests; possibly caused by setting the `Locked` administrative state (see above) or if an error occurred when mounting or dismounting a cartridge on the drive.
- `Broken` will appear only when the PVL is in the midst of shutting itself down because of a fatal error.

- `Unknown` indicates the state of the device is not known to SSM; this is usually caused by the PVL being down or disconnected from SSM.

Usage State

Indicates the state of the PVL's control over the drive. Possible values for this field are:

- `Active` indicates the drive is in use.
- `Idle` indicates the drive is not in use.
- `Busy` indicates the drive is busy.
- `Unknown` indicates the state is not known to SSM.

Drive State

The current state of the drive as reported by the PVL. Possible values for this field:

- `Clear` indicates the drive has been created, but the associated Mover and PVR (if tape) haven't successfully been notified.
- `In Use` indicates a volume is currently mounted or in the process of being mounted on the drive.
- `Free` indicates the drive is unoccupied.
- `Dismount Pending` indicates the drive is in the process of dismounting a volume.
- `Deleted` indicates the drive has been deleted (metadata no longer exists).

Drive Flag

This field describes possible actions that the PVL needs to perform for the drive. Possible flags are:

- `Clear` indicates nothing needs to be done.
- `Modified` is set by SSM when the drive has been modified. It was added for a future feature; currently unsupported.
- `PVR/Mover Notify Pending` indicates the PVL needs to notify the associated PVR and Mover that the drive has been created or deleted.
- `PVR Notify Pending` indicates the PVL needs to notify the associated PVR that the drive has been created or deleted.
- `Mover Notify Pending` indicates the PVL needs to notify the associated Mover that the drive has been created or deleted.
- `Abort PVR/Mover Notify` indicates the PVL is aborting a pending notification.

Controller ID

An indication of the adapter/bus that is in the data path to the device. This field is used to attempt to mount individual volumes of a set of striped media on different controllers, when possible, to prevent contention on the data path.

Polling Interval (tape only)

The number of seconds to wait between polling requests performed by the PVL to determine if any media is present in the drive. This field is disabled by default. Use "-1" to disable polling.

Values of "0" to "14" are not valid. If the drive is an operator-mounted tape drive, then this field shouldn't be disabled.



Advice

It is recommended that this field be set to a value greater than 14 seconds for operator-mounted tape drives and disabled for library managed tape drives. In normal operations for library managed tape drives (for example, STK SL8500), drive polling should not be required. In reality there are occasions where polling may detect a non-HPSS cartridge or an unaccounted for HPSS cartridge. However, the use of the drive polling (originally in place for Operator PVRs) is associated with high overhead (one thread per drive that is polling). A less expensive alternative that addresses this condition is currently being pursued. In the meantime, if you're concerned with monitoring for unexpected cartridges in library managed tape drives , it is recommended that you set the polling interval to a value no less than "900" (15 minutes).

Mounted Volume

The volume that is currently mounted on this drive.

Maintenance Date

The time and date when the drive last had maintenance performed on it. This date must be entered manually and is provided for administrative information purposes only.

Drive Error Count

This field records the number of consecutive Drive Errors (set via the **Retry Mount Time Limit** mechanism) which have occurred on the drive. When the number of errors equal or exceed the PVR **Drive Error Limit**, the drive will automatically be locked by the PVL. The field can be reset to zero administratively using the **Reset** button. A successful mount or PVL recycle will also zero the field.

Mounts Since Last Maintenance

The number of times a volume has been mounted on this drive since the last maintenance. Clicking on the **Reset** button to the right of this field will reset the value to zero.

Comment

This field provides a 128-character buffer in the PVL drive metadata which gives the administrator the opportunity to associate miscellaneous text with a device/drive. For example, a site may want to place a comment in this field that the drive is out of service or being used by another system.

Library Unit ID (tape only)

Since library architecture varies, the method to associate related drives at a unit level will be determined through the use of a "unit id" that is placed in the drive metadata. This will allow the administrator to analyze their specific library configuration and apply their strategy as they feel fit. The use of "unit id" allows the library specifics to be abstracted from the PVL. Library details are the domain of the PVR. See **Defer Dismount Exempt Count** (PVR Server Configuration).

Related information

18.3. Changing device and drive state

The administrative state of a device or drive can be set to Unlocked or Locked. This controls whether HPSS can access the drive. Changing the state of a device or drive can be accomplished via the *Devices and Drives* list window.

Notice that there are two sets of **Lock**, **Unlock** and **Mark Repaired** button groups on the *Devices and Drives* window. The first group is titled **Device Administration** and the second group is titled **Drive Administration**. The first group is working at the Mover device level and the second group is working at the PVL drive level.

Locking *devices* serves no functional purpose. When the device is locked, the information is not transferred to the PVL and thus the PVL will continue to schedule the associated PVL drive. It is also not transferred to the Core Server which will continue to read and write the volume (see **VV Condition** in the *Core Server Volume Disk/Tape* window) that is mounted in the drive.

When a *drive* is locked, the PVL will no longer schedule the PVL drive. When locking a tape drive due to a cartridge problem (such as for a stuck tape), it is beneficial to also cancel the PVL job associated with the cartridge. Additionally, locking the drive is not transferred to the Core Server, which will continue to read and write the volume (see **VV Condition** in the *Core Server Volume Disk/Tape* window) unless you change the **VV Condition** to DOWN (or EOM) which is advised while diagnosing the stuck tape.

18.3.1. Unlocking a drive

The **Administrative State** of a drive must be "unlocked" to allow HPSS to make use of a device/drive. Before unlocking a drive, ensure that its hardware is functional and is fully configured into its host's operating system and into HPSS. Configuration of an HPSS drive is described in Configure a new device and drive.

From the *Devices and Drives* window select the desired device/drive entries and then click on the **Unlock** button from the Drive button group. The drive can also be unlocked by bringing up the *PVL Drive Information* window and setting its **Administrative State** to `Unlocked`.

18.3.2. Locking a drive

To lock a drive, go to the *Devices and Drives* window and select the desired device/drive entries and then click on the **Lock** button from the Drive button group. The drive can also be locked by bringing up the *PVL Drive Information* window and setting its **Administrative State** to `Locked`.

Locking a drive disallows its use by HPSS. Changing a drive state to locked will ensure that the drive will not be used for new mounts.

Always lock the PVL drive instead of the Mover device. Mover devices do not usually need to be locked or unlocked, and the capability to lock and unlock them is provided for troubleshooting purposes only.

Locking a tape drive will not affect an active job which has a volume mounted. Once that job has completed, the drive will be dismounted and no further mounts will take place. This may be useful when preventative maintenance is required for an operating drive.

Locking a disk drive has little effect since disks are logically mounted when the PVL initializes and are not usually dismounted; however, a disk drive must be in the locked state to be deleted.

18.3.3. Repairing the state of a device or drive

A drive can enter an error or suspect state as reported by the PVL, Mover, or both. After a drive has entered one of these abnormal states, it can be repaired to return it to a normal state.

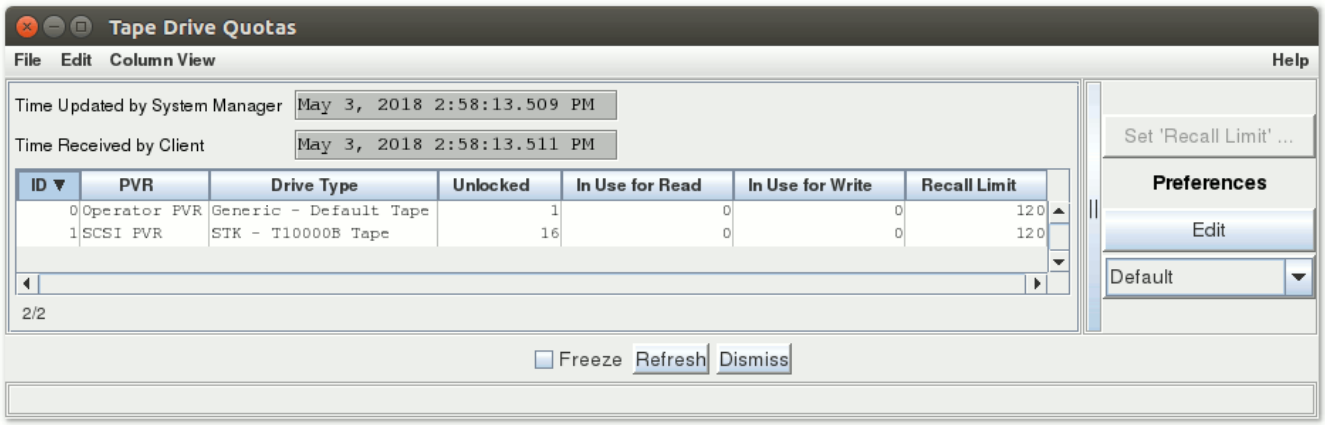
From the *Devices and Drives* window select the desired device/drive entries and then click on the **Mark Repaired** button appropriate for that device/drive type. Another way to change a device/drive state to "repaired" is to bring up the *PVL Drive Information* window or the *Mover Device Information* window and changing the Administrative State to `Mark Repaired`.

Repairing the state of a device/drive is only an instruction to the server to reset the displayed device/drive state value. It does not correct any underlying problems that might still exist. Rather, it is a means by which the administrator can notify the server that the underlying problem has been addressed outside of the server. It is used for problems that the server cannot fix by itself such as a drive being offline or a tape getting stuck in a drive.

18.3.4. Resetting drive statistics

Both the PVL and Mover maintain statistics related to a drive. The PVL maintains an error count as well as a count of the number of times a cartridge is mounted. The Mover maintains the count of the number of errors encountered and the number of bytes read and written on the drive. These values can all be reset to zero by the **Reset** button located next to the statistic field on the appropriate window.

18.3.5. Tape Drive Quota list window



This window allows you to manage the tape drive recall limit by PVR and tape drive type.

Field descriptions

ID

The ID associated with this device/drive.

PVR

The descriptive name of the PVR.

Drive Type

The tape drive type.

Unlocked

The number of tape drives of this type in this PVR that are currently unlocked.

In Use for Read

The number of tape drives of this type in this PVR that are currently in use for read activity.

In Use for Write

The number of tape drives of this type in this PVR that are currently in use for write activity.

Recall Limit

The maximum number of drives of this type in this PVR that can be used at the same time for data recall. A value of -1 disables this feature.

Chapter 19. Volume and storage management

This chapter describes the procedures for adding, removing, monitoring, and managing storage space in the HPSS system.

The basic unit of storage which can be added to the HPSS system is the volume. Before volumes can be added to HPSS, the underlying configuration structures must be created to support them. These structures include:

- Storage classes, hierarchies, classes of service, migration policies, purge policies, and, optionally, file families. See the Storage configuration chapter for the procedures for creating these structures.
- Device configurations. See the Device and drive management chapter for the procedures for creating these structures.

19.1. Adding storage space

Adding storage space to the HPSS system means creating new virtual volumes and adding them to a storage class. Creating new virtual volumes in a storage class is a two-step process:

1. **Import volumes.** In the first step, the physical volumes that will be gathered into the new Core Server virtual volumes are imported by the PVL.
2. **Create resources.** In the second step, metadata for the new volumes is created by the Core Server; this process is called creating storage resources.

19.1.1. Importing volumes into HPSS

Importing a physical volume (PV) into HPSS is the process of adding a tape cartridge, disk volume, or object store volume to the PVL and labeling it with an HPSS volume label (disk and tape). Importing also makes the volume known to the PVR and PVL, and creates the appropriate metadata records. Cleaning cartridges are managed by the tape library directly and should not be imported into the HPSS system.



To import a tape cartridge into HPSS, the administrator must be familiar with the operation of the PVR to which the cartridge will be added because the physical cartridge injection process differs among PVRs.

- **Operator** - No cartridge injection step is necessary (or possible).
- **SCSI** - When importing new cartridges into HPSS, the cartridges must be entered into an input/output (I/O) slot before any HPSS import operations are performed. The HPSS SCSI PVR moves the cartridges into an available slot while importing into HPSS.
- **STK** - When importing new cartridges into HPSS, the cartridges must be entered into the STK robot before any HPSS import operations are performed. See the *STK Automated Cartridge*

System Library Software (ACSLS) System Administrator's Guide for procedures to enter cartridges.

Cartridges may be entered into the STK silo through various means, including:

- Using the enter command of ACSLS (StorageTek's robot software) and putting the cartridges into the Cartridge Access Port (CAP).
- Loading a cartridge into a CAP operating in ACSLS's "hot enter" mode.
- Bulk loading a robot and then performing an ACSLS audit of the slots that have been filled.
- STK supports cartridges without bar code external labels. These cartridges can be entered into the robot using the *venter* command at the ACSLS console. These cartridges are fully supported by HPSS, but the robot will eject the cartridges if they are scanned during an audit.

We strongly recommend that tape cartridges be imported into a robot that verifies external labels before mounting in a drive. This will ensure that a blank tape being imported has the appropriate external label before it is labeled. Once labeled, a cartridge can be moved to another PVR, such as an operator (hand) mounted PVR.

If labels are written on tape volumes prior to importing those volumes into HPSS, then care should be taken that the label has a format that HPSS can process. In particular, the first four characters of the OwnerID field should be "HPSS" and the last two characters of this fourteen character field should be "00". HPSS uses the last two characters to indicate the side number of the cartridge in anticipation of supporting cartridges that have multiple physical sides; for example, optical disk cartridges that contain an A and a B side. If these two characters are not set correctly, then the import will fail because the side information contained in the internal label will not match the side information passed to PVL in the Import request. If the start of the OwnerID field is not "HPSS", then the volume will be imported as a Foreign Label volume and the side will be set to zero.

When importing a non-removable disk volume into HPSS, the raw disk must already be defined in the host system. The import labels the volumes with HPSS labels.

The volumes can be imported into HPSS using the *Import Disk Volumes* window, the *Import Object Store Volumes* window, or the *Import Tape Volumes* window in the following sections. Once the volumes are imported, the HPSS storage space can be created using the *Create Disk Resources* window or the *Create Tape Resources* window.

The import step must be done for disk, tape, and object store volumes. Ensure that the PVL, PVR, and appropriate Movers are up and running before initiating a requesting for the import operation. Also ensure that SSM is connected to the PVL.

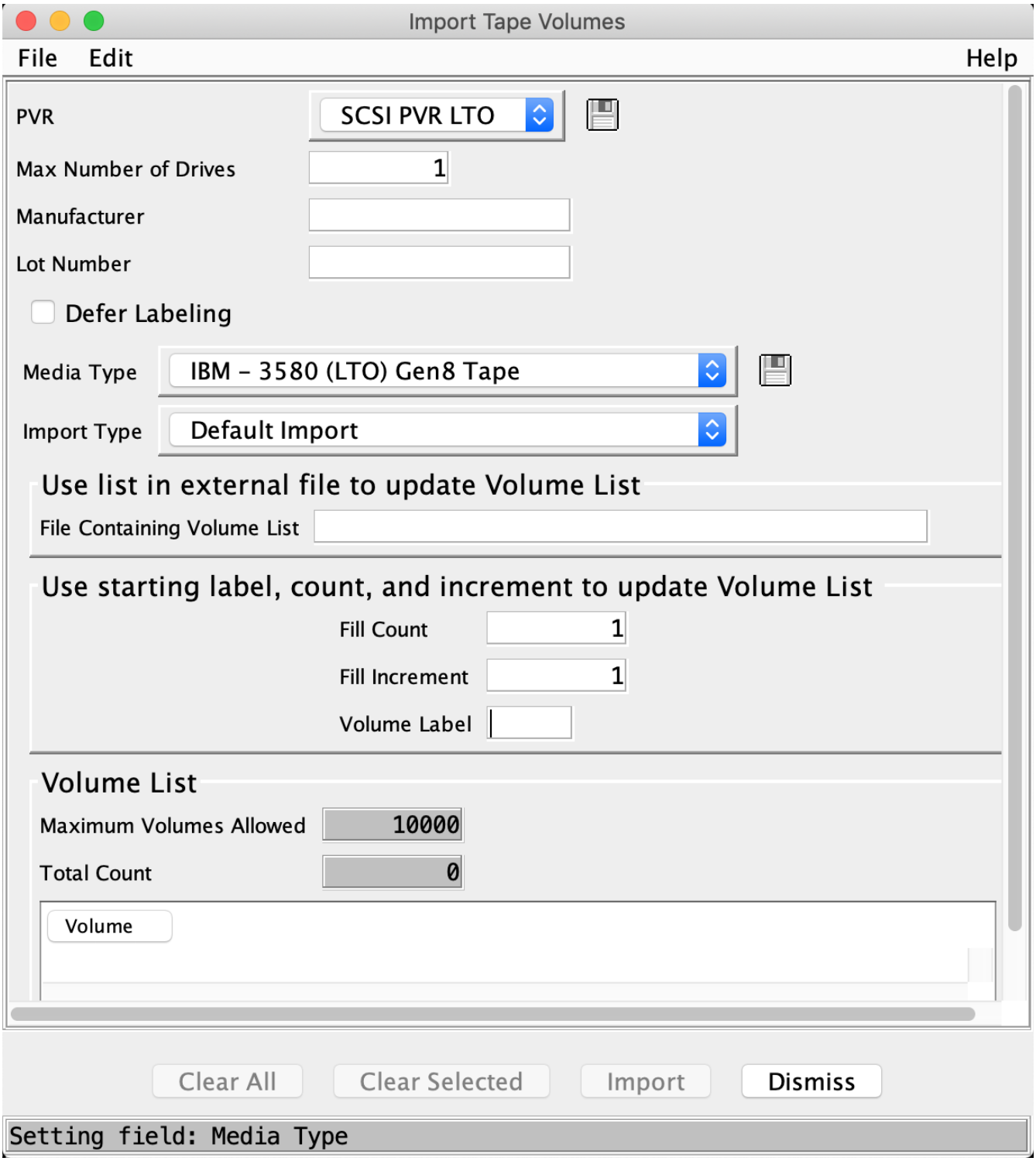
When importing tapes, the maximum number of tape drives used by the import process is controlled by the **Max Number of Drives** field in the *Import Tape Volumes* window. Each tape must be mounted and its label must be written or checked. The PVL will attempt to evenly distribute the work across the specified number of tape drives. The administrator should consider the current and expected demand for tape drives of the type needed for the import, when setting this field.

When imports to an operator PVR are performed, requests to mount the tape cartridges are serialized. This is necessary to ensure proper labeling of unlabeled tapes. Any unlabeled cartridge

mounted in a drive under control of the operator will be assumed to be the requested cartridge. Because any tape that is mounted will be labeled without machine checking of the external label, this operation is not without risk. It is strongly recommended that labeling of cartridges be performed by a robot that can verify the external label of a cartridge, if possible.

When SSM completes the import request, it reports the number of volumes imported in the window status field. Any error causes SSM to stop processing the request. If the reported number is less than the number requested, you should retry the import request after first addressing the reason for the initial failure. For the retry, the original list of volumes to import can be reused because the PVL will skip volumes that have already been imported.

19.1.1.1. Import Tape Volumes window



This window allows the user to import tape volumes into the HPSS system, making them known to the PVL server. To make them known to the Core Server so they can be used by HPSS, storage resources must then be created for the volumes via the *Create Tape Resources* window.

When the **Max Number of Drives** is set to "1", the volumes are processed one at a time in sequence. If an error occurs, processing stops. The window completion message will report the number of successful imports, from which the volume causing the failure can be found. For example, if you requested import of 100 volumes, and the completion message showed that 37 imports were successful, then the 38th volume in the **Volume List** caused the failure.

When the **Max Number of Drives** is greater than "1", the SSM System Manager effectively starts multiple import jobs (equal to **Max Number of Drives**) which run at the same time. Each job independently processes part of the **Volume List**, and stops when its part of the list is done or when an error occurs. The window completion message shows only the total of all successful imports from all of these jobs. It is therefore impossible to tell from this number which volume caused the failure. In this case it is important to review the *Alarms and Events* window to get more information on the failures.

If you request the import of a volume which was previously imported, the System Manager counts this as a successful import and goes on to the next volume in the list. This makes it easy to restart a partially completed import (after fixing the cause of the error which terminated the first request) by clicking the **Import** button again. There is no need to remove from the list the volumes which were imported successfully.

The list of the volumes to be imported may be constructed in any of three ways. Each volume name may be typed in one at a time, or a list of volume names may be automatically generated from a single entry, or a list of volume names may be specified from an input file. Volume names are not typed directly into the **Volume List** at the bottom of the window.

Any of the three entry methods may be repeated multiple times on the same window to add additional volumes to the list. All three entry methods or any combination of them may be used in succession on the same window.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"
```

```
Labels automatically inserted into Volume List:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
"AB7329"
```

The filling will not occur and an error will be displayed if the specified values would generate an invalid volume label (for example, one greater than zzz999).

To specify a list of volumes from a file, create a file containing the name of each volume to be imported on a separate line. Volume names must be six alphanumeric characters. No other characters are allowed on the line. The file must be accessible from the host on which the hpssgui is executing. Enter the name of the file in the **File Containing Volume List** field. The volume names from the file will be added to the **Volume List**.

Field descriptions

PVR

The descriptive name of the PVR into which the tape will be imported.

Max Number of Drives

The maximum number of tape drives to use when running the tape imports. Valid values are any positive integer value, up to the number of drives available, to a maximum of 256. The default value is "1".

Manufacturer

The maker of the media to be imported. Any character string is valid. The field may be left blank.

Lot Number

The manufacturer's lot number for the media. Any character string is valid. The field may be left blank.

Defer Labeling

If selected, tape(s) in this import will not be labeled and will instead be labeled when they are first used for a write.

Media Type

The type of media to be imported.

Import Type

The type of import to be performed. Valid values are `Default Import`, `Scratch Import`, and `Overwrite Import`. The default value is `Default Import`. Specifying a `Scratch Import` type will usually cause an HPSS label to be written onto the media, potentially causing any data already on the media to be lost. Specifying an `Overwrite Import` type will cause the HPSS label to be rewritten, if an existing HPSS or Foreign label already exists, provided it is for the same Volume ID. This allows tapes to be rewritten in newer drives (for example, tapes originally created on a 3590 drive to be reused on a 3590E drive). Based on the **Import Type**, the import request will be processed depending on how the media is currently labeled. See Selecting import type for tape cartridges.

for more information on selecting the appropriate import type.

File Containing Volume List

The name of an external file (accessible from the host on which the hpssgui(7) is executing) containing a list of volume labels to be added to the end of the **Volume List**.

Fill Count

The number of labels to be added to the **Volume List** when a value is typed into the **Volume Label** field. Valid values are positive integers greater than or equal to "1". The default value is "1". If the value specified for **Fill Count** would cause the list to grow beyond the Maximum Volumes Allowed, an error message is displayed (see **Maximum Volumes Allowed** below).

Fill Increment

This field determines how each new volume label is generated. The **Fill Increment** is the number by which each automatically generated label will differ from the previous one when a value is typed into the **Volume Label** field and the **Fill Count** is greater than "1". Valid values are any positive integer. The default value is "1". The **Fill Increment** is added to the least significant part of a volume label to generate a new label.

Volume Label

The six-character alpha-numeric label of a volume to be imported. The label is added to the end of the **Volume List**. If **Fill Count** is greater than "1", multiple labels are generated using the entered label as a starting point.

Maximum Volumes Allowed

The maximum number of volume labels that will fit in the **Volume List**. The value is 10,000 and

is set by SSM. This field is non-editable.

Total Count

The total number of tapes to be imported. This is an informational field reflecting the number of volume names generated in the **Volume List** and is not directly editable.

Volume List

A list of volume labels specifying the volumes to be imported. You cannot enter labels directly into this list, but must construct it in one of the three ways described above. Use the scrollbar to move up and down the list of labels.

You can select one or more labels from the list for the purpose of deleting them from the list (see **Clear Selected** below). To select one label, click on it with the mouse; the selection will be highlighted. To select a range, select one label; then select a second label while holding down the Shift key. The selected labels, and all labels between the two, will be highlighted. You may also hold down the Control key while selecting, to select or deselect individual labels without affecting any of the others.

Buttons

Clear All

Clears the **Volume List**, and resets **Fill Count** and **Fill Increment** to "1".

Clear Selected

If one or more volumes are selected (highlighted) in the **Volume List**, clicking on this button will remove them from the list. Note that this does not actually delete anything from HPSS.

Import

Begins the volume import using the displayed data. A start message is displayed on the message line at the bottom of the window, and all window features are disabled, except for the **Dismiss** button; this prevents you from modifying any data on the window while the import is in progress.

It is a good idea to have the *Alarms and Events* window open while import proceeds, so that all relevant log messages can be viewed. For tape imports, it is also helpful to have the *Tape Mount Requests* window open, where you can see mount requests for each volume come and go as they are imported.

When the import is finished, a completion message is displayed and the window features are sensitized again. You may dismiss the window before completion; however, completion messages will be displayed in a pop-up window. At this point you can begin entering data for another import, or you can dismiss the window.

19.1.1.2. Selecting import type for tape cartridges

The following table lists information for selecting tape import types.



Any existing data on a volume imported into HPSS will be overwritten and subsequently lost.

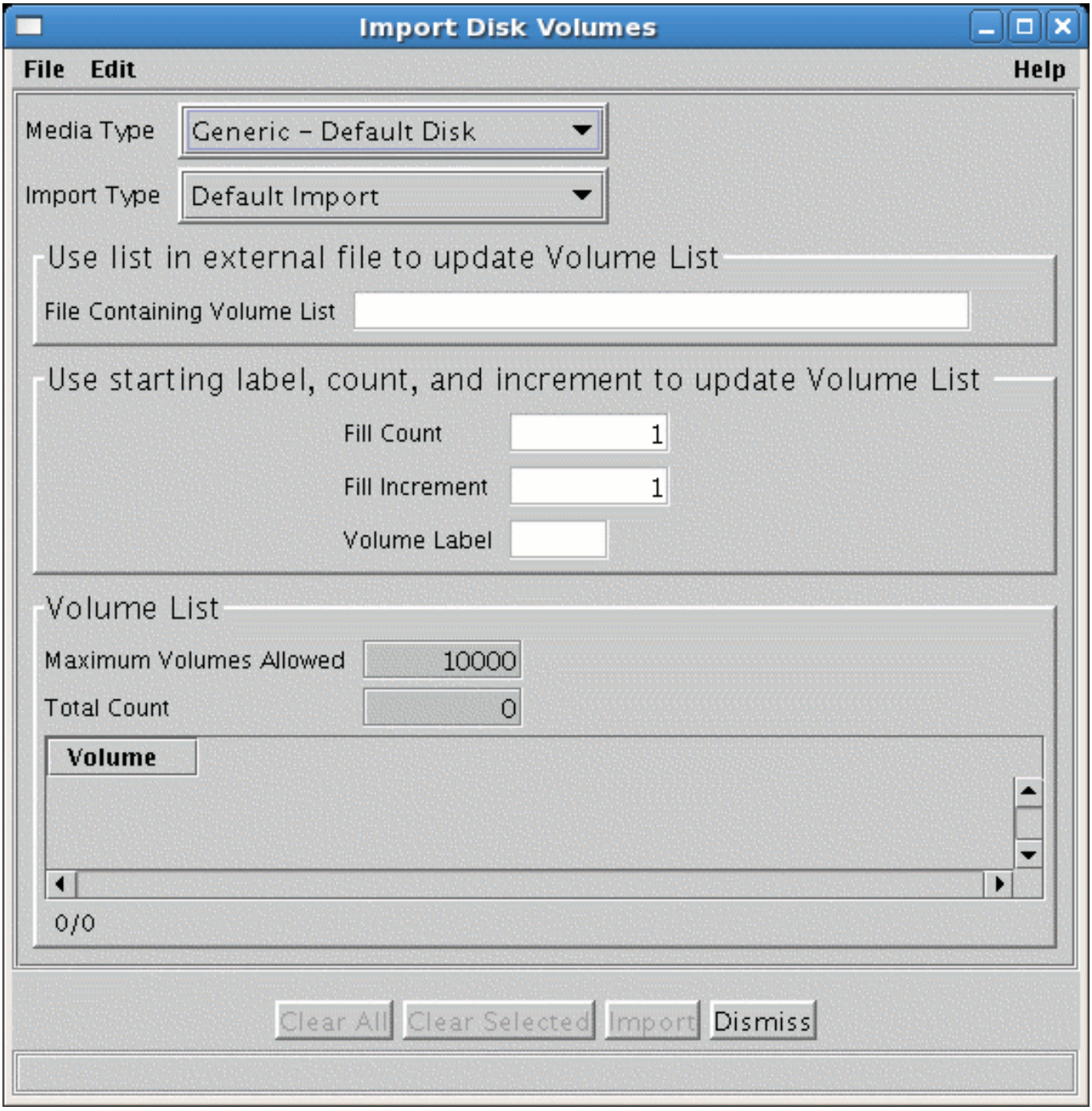
Table 28. Tape import types

Current tape label	Default import	Overwrite import	Scratch import
A UniTree label (an ANSI label without a trailing tape mark) with a correct Volume ID	Tape imported	Label written, tape imported	Tape imported
An ANSI (non-UniTree) or HPSS label with a correct Volume ID (the Volume ID on the label is as expected by HPSS)	Tape imported	Label written, tape imported	Tape imported
An ANSI or HPSS label with an incorrect Volume ID (the Volume ID on the label is different from the Volume ID expected by HPSS)	Tape not imported	Tape not imported	Tape not imported
Random data (for example, a tar file)	Tape not imported	Label written, tape imported	Label written, tape imported
No data (two tapemarks at the start of tape)	Label written, tape imported	Label written, tape imported	Label written, tape imported
Unreadable (for example, some brand new tapes or a degaussed tape; also possibly a tape written at a different density)	Tape not imported	Label written, tape imported	Label written, tape imported

HPSS always attempts to read a label at the beginning of a tape when performing an import. Some tape drives and device drivers will report errors when asked to read unreadable tapes (as described above). If this happens, manually write two tape marks at the start of the tape and retry the import. Most UNIX systems provide the `mt` command, which can be used to write tape marks.

An HPSS label is basically just an ANSI label. A few characters are changed to identify the tape as having been labeled by HPSS. Both label types are supported by HPSS; tapes already beginning with an ANSI label are not relabeled unless the overwrite import option is selected and volume labels match.

19.1.1.3. Import Disk Volumes window



This window allows the user to import disk volumes into the HPSS system, making them known to the PVL and PVR servers. To make them known to the Core Server so they can be used, they must be created via the Core Server's *Create Disk Resources* window.

The SSM System Manager processes the volumes one at a time in sequence. If it encounters an error, it stops and returns. The window completion message will report the number of successful imports, from which the volume causing the failure can be found. For example, if you requested import of 100 volumes, and the completion message showed that 37 imports were successful, then the 38th volume in the **Volume List** caused the failure.

If you request the import of a volume which was previously imported, the System Manager counts this as a successful import and goes on to the next volume. This makes it easy to restart a partially completed import (after fixing the cause of the error which terminated the first request) by clicking the **Import** button again. There is no need to remove the already imported volumes from the list.

The list of the volumes to be imported may be constructed in any of three ways. Each volume name may be typed in one at a time, or a list of volume names may be automatically generated from a single entry, or a list of volume names may be specified from an input file. Volume names are not typed directly into the **Volume List** at the bottom of the window.

Any of the three entry methods may be repeated multiple times on the same window to add additional volumes to the list. All three entry methods or any combination of them may be used in succession on the same window.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
```

```
"AB7329"
```

The filling will not occur and an error will be displayed if the specified values would generate an invalid volume label (for example, one greater than zzz999).

To specify a list of volumes from a file, create a file containing the name of each volume to be imported on a separate line. Volume names must be six alphanumeric characters. No other characters are allowed on the line. The file must be accessible from the host on which the hpssgui(7) is executing. Enter the name of the file in the **File Containing Volume List** field. The volume names from the file will be added to the **Volume List**.

Field descriptions

Media Type

The type of media to be imported.

Import Type

The type of import to be performed. Valid values are `Default Import`, `Scratch Import`, and `Overwrite Import`. Based on the import type, the import request will be processed depending on how the media is currently labeled. See Selecting import type for disk volumes.

for more information on selecting the appropriate import type.

File Containing Volume List

The name of an external file containing a list of volume labels to be added to the end of the **Volume List**.

Fill Count

The number of labels to be added to the **Volume List** when a value is typed into the **Volume Label** field. Valid values are positive integers greater than or equal to "1". The default value is "1". If the value specified for **Fill Count** would cause the list to grow beyond the **Maximum Volumes Allowed**, an error message is displayed (see **Maximum Volumes Allowed** below).

Fill Increment

This field determines how each new volume label is generated. The **Fill Increment** is the number by which each automatically generated label will differ from the previous one when a value is typed into the **Volume Label** field and the **Fill Count** is greater than "1". Valid values are any positive integer. The default value is "1". The **Fill Increment** is added to the least significant part of a volume label to generate a new label.

Volume Label

The six-character alpha-numeric label of a volume to be imported. The label is added to the end of the **Volume List**. If **Fill Count** is greater than "1", multiple labels are generated using the entered label as a starting point.

Maximum Volumes Allowed

The maximum number of volume labels that will fit in the **Volume List**. The value is 10,000 and is set by SSM. This field is non-editable.

Total Count

The total number of disks to be imported. This is an informational field reflecting the number of volume names generated in the **Volume List** and is not directly editable.

Volume List

A list of volume labels specifying the volumes to be imported. You cannot enter labels directly into this list, but must construct it using one of the three ways described above. Use the scrollbar to move up and down the list of labels.

You can select one or more labels from the list for the purpose of deleting them from the list (see **Clear Selected** below). To select one label, click on it with the mouse; the selection will be highlighted. To select a range, select one label; then select a second label while holding down the Shift key. The selected labels, and all labels between the two, will be highlighted. You may also hold down the Control key while selecting, to select or deselect individual labels without affecting any of the others.

Buttons

Clear All

Clears the **Volume List**, and resets **Fill Count** and **Fill Increment** to "1".

Clear Selected

If one or more volumes are selected (highlighted) in the **Volume List**, clicking on this button will remove them from the list. Note that this does not actually delete anything from HPSS.

Import

Begins the volume import using the displayed data. A start message is displayed on the message line at the bottom of the window, and all window features are disabled, except for the **Dismiss** button; this prevents you from modifying any data on the window while the import is in progress.

It is a good idea to have the *Alarms and Events* window open while import proceeds, so that all relevant log messages can be viewed.

When the import is finished, a completion message is displayed and the window features are sensitized again. You may dismiss the window before completion; however, completion messages will be displayed in a pop-up window. At this point you can begin entering data for another import, or you can dismiss the window.

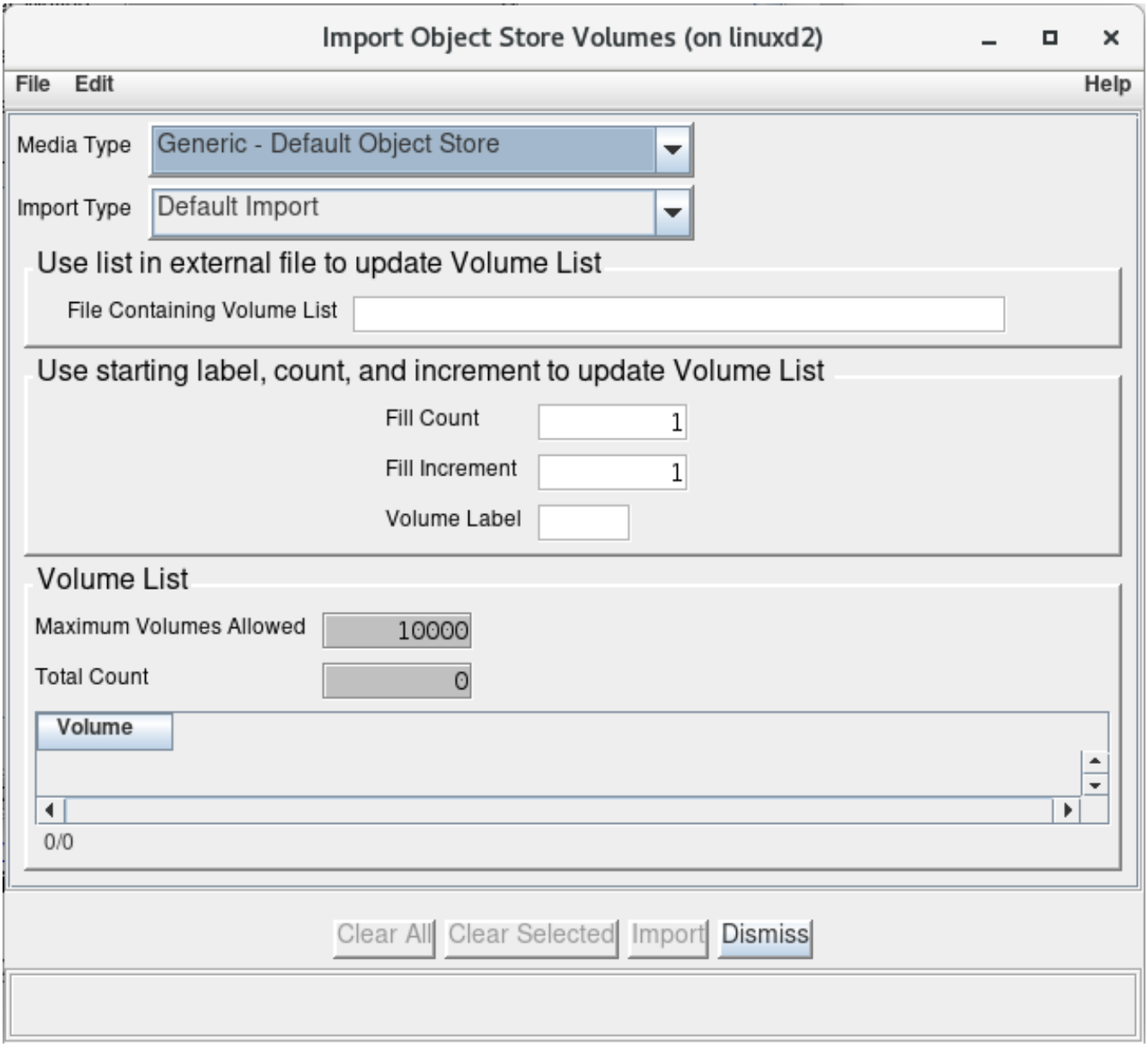
19.1.1.4. Selecting import type for disk volumes

The following table lists information for selecting disk import types.

Table 29. Disk import types

Current disk label	Default import	Overwrite import	Scratch import
An ANSI or HPSS label with a correct Volume ID (the Volume ID on the label is as expected by HPSS)	Disk imported	Label written, disk imported	Disk imported
An ANSI or HPSS label with an incorrect Volume ID (the Volume ID on the label is different from the Volume ID expected by HPSS)	Disk not imported	Disk not imported	Disk not imported
No label	Label written, disk imported	Label written, disk imported	Label written, disk imported

19.1.1.5. Import Object Store Volumes window



This window allows the user to import object store volumes into the HPSS system, making them

known to the PVL and PVR servers. To make them known to the Core Server so they can be used, they must be created via the Core Server's *Create Object Store Resources* window.

The SSM System Manager processes the volumes one at a time in sequence. If it encounters an error, it stops and returns. The window completion message will report the number of successful imports, from which the volume causing the failure can be found. For example, if you requested import of 100 volumes, and the completion message showed that 37 imports were successful, then the 38th volume in the **Volume List** caused the failure.

If you request the import of a volume which was previously imported, the System Manager counts this as a successful import and goes on to the next volume. This makes it easy to restart a partially completed import (after fixing the cause of the error which terminated the first request) by clicking the **Import** button again. There is no need to remove the already imported volumes from the list.

The list of the volumes to be imported may be constructed in any of three ways. Each volume name may be typed in one at a time, or a list of volume names may be automatically generated from a single entry, or a list of volume names may be specified from an input file. Volume names are not typed directly into the **Volume List** at the bottom of the window.

Any of the three entry methods may be repeated multiple times on the same window to add additional volumes to the list. All three entry methods or any combination of them may be used in succession on the same window.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "000070"

Labels automatically inserted into Volume List:
"000070"
"000080"
"000090"
"000100"
"000110"
"000120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "007329"

Labels automatically inserted into Volume List:
"007329"
"009329"
"011329"
"013329"
"015329"
"017329"
```

The filling will not occur and an error will be displayed if the specified values would generate an invalid volume label (for example, one greater than zzz999).

To specify a list of volumes from a file, create a file containing the name of each volume to be imported on a separate line. Volume names must be six alphanumeric characters. No other characters are allowed on the line. The file must be accessible from the host on which the hpssgui(7) is executing. Enter the name of the file in the **File Containing Volume List** field. The volume names from the file will be added to the **Volume List**.

Field descriptions

Media Type

The type of media to be imported.

Import Type

The type of import to be performed. Valid values are `Default Import`, `Scratch Import`, and `Overwrite Import`.



This field is currently not used for object store volume imports.

File Containing Volume List

The name of an external file containing a list of volume labels to be added to the end of the **Volume List**.

Fill Count

The number of labels to be added to the **Volume List** when a value is typed into the **Volume Label** field. Valid values are positive integers greater than or equal to "1". The default value is "1". If the value specified for **Fill Count** would cause the list to grow beyond the **Maximum Volumes Allowed**, an error message is displayed (see **Maximum Volumes Allowed** below).

Fill Increment

This field determines how each new volume label is generated. The **Fill Increment** is the number by which each automatically generated label will differ from the previous one when a value is typed into the **Volume Label** field and the **Fill Count** is greater than "1". Valid values are

any positive integer. The default value is "1". The **Fill Increment** is added to the least significant part of a volume label to generate a new label.

Volume Label

The six-character alpha-numeric label of a volume to be imported. The label is added to the end of the **Volume List**. If **Fill Count** is greater than "1", multiple labels are generated using the entered label as a starting point.

Maximum Volumes Allowed

The maximum number of volume labels that will fit in the **Volume List**. The value is 10,000 and is set by SSM. This field is non-editable.

Total Count

The total number of object store volumes to be imported. This is an informational field reflecting the number of volume names generated in the **Volume List** and is not directly editable.

Volume List

A list of volume labels specifying the volumes to be imported. You cannot enter labels directly into this list, but must construct it using one of the three ways described above. Use the scrollbar to move up and down the list of labels.

You can select one or more labels from the list for the purpose of deleting them from the list (see **Clear Selected** below). To select one label, click on it with the mouse; the selection will be highlighted. To select a range, select one label; then select a second label while holding down the Shift key. The selected labels, and all labels between the two, will be highlighted. You may also hold down the Control key while selecting, to select or deselect individual labels without affecting any of the others.

Buttons

Clear All

Clears the **Volume List**, and resets **Fill Count** and **Fill Increment** to "1".

Clear Selected

If one or more volumes are selected (highlighted) in the **Volume List**, clicking on this button will remove them from the list. Note that this does not actually delete anything from HPSS.

Import

Begins the volume import using the displayed data. A start message is displayed on the message line at the bottom of the window, and all window features are disabled, except for the **Dismiss** button; this prevents you from modifying any data on the window while the import is in progress.

It is a good idea to have the *Alarms and Events* window open while import proceeds, so that all relevant log messages can be viewed.

When the import is finished, a completion message is displayed and the window features are sensitized again. You may dismiss the window before completion; however, completion messages will be displayed in a pop-up window. At this point you can begin entering data for

another import, or you can dismiss the window.

19.1.2. Creating storage resources

Creating storage resources is the process of creating Core Server metadata to manage the space on HPSS volumes. The volumes must be imported into HPSS (see Importing volumes into HPSS).

before resources may be created on them.

Before storage resources can be created, the administrator must determine which storage subsystem will receive the new storage, and which storage class they will be placed in. The choice of storage subsystem determines the Core Server that will be used.

Ensure that the PVL and the appropriate Core Server are running before initiating a request to create storage resources. Also ensure that SSM is connected to the appropriate Core Server.

The selected Core Server will create a number of HPSS virtual volumes. All new tape virtual volumes will have identical characteristics (including type, estimated size, stripe size, stripe width, block size, and blocks between tape marks). When creating disk storage resources, the administrator may override the default PV Size and enter a value specific to the circumstance.

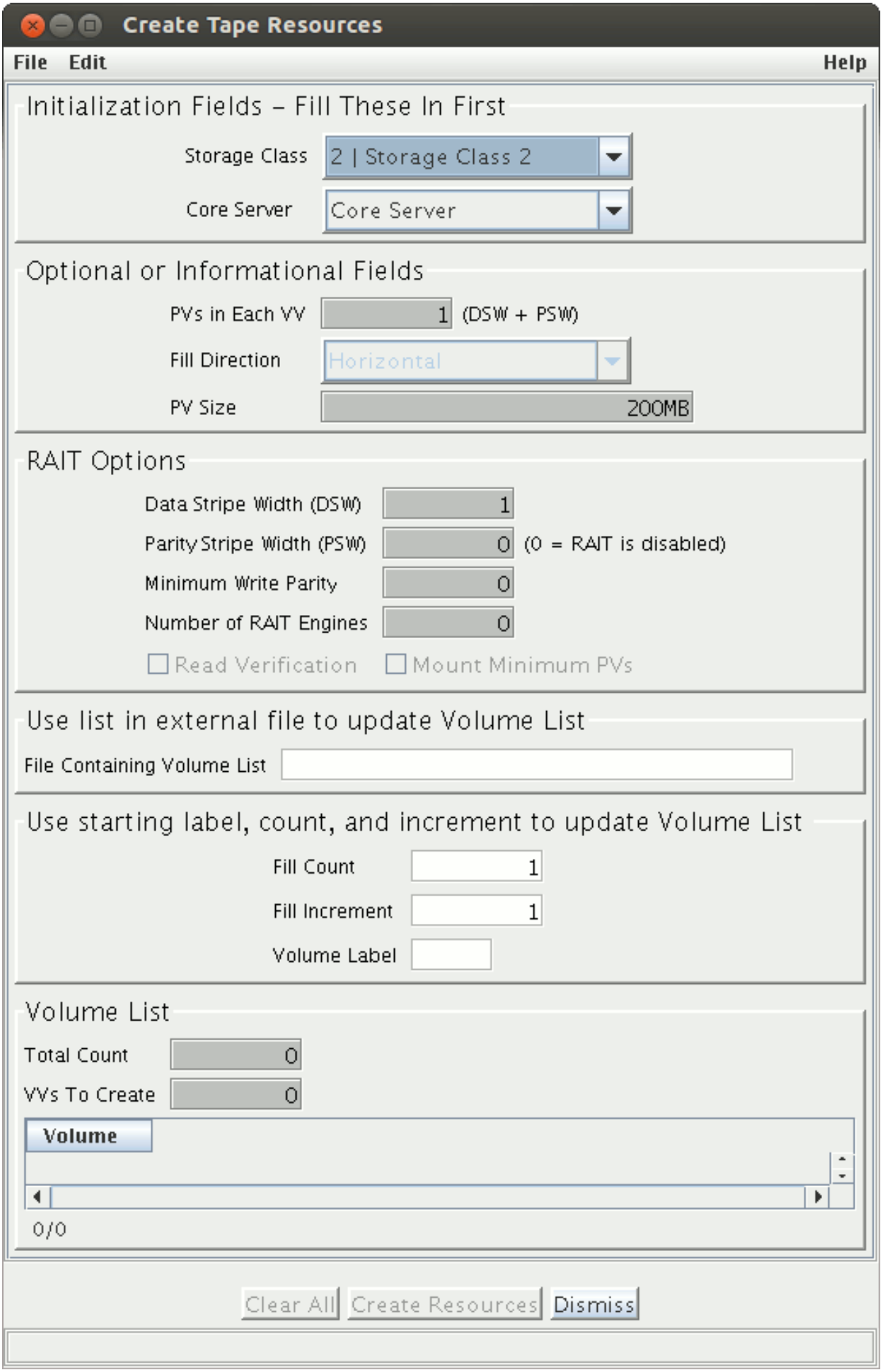
Virtual volumes are created by the Core Server by grouping physical volumes together in groups of N, where N is the total stripe width of the resulting virtual volume. The physical volumes are arranged into virtual volumes according to the table in the resource creation window, which the administrator should modify as desired before clicking the **Create** button. Every physical volume is placed into a virtual volume before it is usable by HPSS, even if there is only one physical volume per virtual volume.

It is important to understand that the storage class characteristics in effect when volumes are created are assigned to the new volumes. Once volumes are created, changes to the storage class have no effect on the characteristics of existing volumes. The storage class characteristics (such as block sizes and stripe widths) are copied to the volumes' metadata and become a permanent part of the definition of the volumes.

The Core Server creates the necessary metadata structures for each of the new virtual volumes. Each new virtual volume is immediately available for use.

Note that new tape resources are not assigned to a file family. Tapes are assigned to file families from the unused tapes in the storage class as they are needed to satisfy requests to write to tape in a given family. Tapes cannot be pre-assigned to file families.

19.1.2.1. Create Tape Resources window


Create Tape Resources
File
Edit
Help
Initialization Fields – Fill These In First
Storage Class
2 | Storage Class 2
Core Server
Core Server
Optional or Informational Fields
PVs in Each VV
1
(DSW + PSW)
Fill Direction
Horizontal
PV Size
200MB
RAIT Options
Data Stripe Width (DSW)
1
Parity Stripe Width (PSW)
0
(0 = RAIT is disabled)
Minimum Write Parity
0
Number of RAIT Engines
0
Read Verification
Mount Minimum PVs
Use list in external file to update Volume List
File Containing Volume List
Use starting label, count, and increment to update Volume List
Fill Count
1
Fill Increment
1
Volume Label
Volume List
Total Count
0
VVs To Create
0
Volume
0/0
Clear All
Create Resources
Dismiss

This window is used to create tape storage resources, tape virtual volumes, in a Core Server. The tapes must first be imported to the appropriate PVL.

The names of the volumes may be entered into the window one at a time, or a list of volume names may be automatically generated from a single entry. Volume names are not entered directly into the **Volume List** at the bottom of the window but are typed into the **Volume Label** field.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List table:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
"AB7329"
```

The following shows an example of how the **Fill Direction** field works when set to `Horizontal` and the **PVs in Each VV** value is "4" for the selected storage class.

Example:

```
PVs in Each VV = 4
Fill Direction = Horizontal
Fill Count = 8
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA007000","AA008000","AA009000","AA010000"
"AA011000","AA012000","AA013000","AA014000"
```

The following shows an example of how the **Fill Direction** field works when set to `Vertical` and the **PVs in Each VV** value is "4" for the selected storage class.

Example:

```
PVs in Each VV = 4
Fill Direction = Vertical
Fill Count = 8
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA007000","AA009000","AA011000","AA013000"
"AA008000","AA010000","AA012000","AA014000"
```

Once the Core Server completes the create request, SSM reports the number of physical volumes and virtual volumes created in the window status field.

Field descriptions

Initialization Fields - Fill These In First

Storage Class

The name of the storage class whose characteristics will be used to create the requested virtual volumes. The stripe width of this storage class will determine the number of columns in the **Volume List** table at the bottom of the window.

Core Server

The name of the Core Server that will process the create request and in which the tape storage resources will be created. The **Storage Class** field must be filled in before selecting a Core Server.

Optional or Informational Fields

PVs in Each VV

The **Data Stripe Width** plus **Parity Stripe Width** of the selected storage class. This field may not be changed.

Fill Direction

Determines which direction filling takes place in the table. **Fill Direction** will not be enabled if the **PVs in Each VV** is "1". By default, if the **Fill Direction** field is enabled, it is set to `Horizontal`. Select the desired value for this field before entering a **Volume Label**.

PV Size

The size, in bytes, of each physical volume used. The default value is the value from the storage class. For tapes, this is an estimate and is informational only.

RAIT options

Data Stripe Width

The number of PVs in each new VV that will contain file information. This field may not be changed.

Parity Stripe Width

The number of PVs in each new VV that will contain RAIT parity information. When Parity Stripe Width is greater than zero, the VV is RAIT. When zero, the VV is not RAIT. This field may not be changed.

Minimum Write Parity

The minimum number of parity PVs that must remain writable in order for the RAIT VV to be writable. This field may not be changed.

Number of RAIT Engines

The number or RAIT Engines that will be used when reading or writing the VVs. This field may not be changed.

Read Verification

When set, parity information in the RAIT VV will be used to verify the integrity of data read from the VV. This field can be set or cleared before the VVs are created.

Mount Minimum PVs

When set, the smallest number of PVs necessary will be mounted when reading the RAIT VV. When **Read Verification** is not set, this will be equal to the value of DSW; and when **Read Verification** is set, this will equal (DSW + 1).

Use list in external file to update Volume List

File Containing Volume List

The name of an external file (accessible from the host on which the hpssgui(7) is executing) containing a list of volume labels to be added to the end of the **Volume List**.

Use starting label, count, and increment to update Volume List

Fill Count

The number of labels to be added to the **Volume List** when a value is typed into the **Volume Label** field. Valid values are positive integers greater than or equal to "1" up to the number of drives available. The default value is "1". If the list fills before the **Fill Count** is exhausted, filling stops and a message is displayed.

Fill Increment

This field determines how each new volume label is generated. The **Fill Increment** is the number by which each automatically generated label will differ from the previous one when a value is in the **Volume Label** field and the **Fill Count** is greater than "1". Valid values are positive integers up to the number of available drives. The default value is "1". A volume label is six alphanumeric characters. The **Fill Increment** is added to the least significant portion of a volume label to generate a new label.

Volume Label

The next label that will be generated and inserted into the table. The labels are generated when the field loses focus. Select **Fill Count**, **Fill Increment**, and **Fill Direction** before entering a value in this field.

Volume List

A list of volume labels specifying the volumes used to create resources. You cannot enter labels directly into this list but must instead use the **Fill Count**, **Fill Increment**, and **Volume Label** fields. Use the scrollbar to navigate the list of labels.

Total Count

The total number of physical volumes in the generated Volume List. This is an informational field and not directly enterable.

VVs To Create

The number of virtual volumes that will be created. This is an informational field and not directly enterable.

Buttons

Clear All

This button clears the volume list and **Volume Label** field.

Create Resources

Begins the process of creating HPSS storage resources using the displayed data. A start message is displayed on the status bar, and all window features are disabled, except for the **Dismiss** button. This prevents the user from modifying any data on the window while the create request is in progress.

19.1.2.2. Prioritize Tape Resource Selection

HPSS selects new tape volumes for writing based on their priority. Volumes are selected in order starting at priority 0. The priority of a tape volume can be identified either using the dump_sspvs(7)

tool with the -P option, or by using the hpss_vv_select_priority(7) tool to see the selection priority and criteria for each VV in the storage class. When two or more volumes have the same priority, the volume with the earliest volume creation time will be selected.

When tape volumes are created, they are assigned the same default priority, which is the least possible priority (e.g. 4294967295). Each volume will have the same value, which will cause them to be selected by volume creation time.

The priority of the tape volumes can be changed in two ways. The first method, called the policy-based method, uses the PVR or PVR and drive type combinations. This corresponds to the `hpss_update_priority_by_pvr` and `hpss_update_priority_by_pvr_drive` tools. These tools will organize the tape volumes in the storage class based on a weighing criteria (PVR or PVR/Drive combo). This criteria is applied such that cartridges will be selected in a sequence, balanced by the weights, across the available set of tape volumes (see `hpss_update_priority_by_pvr` for an example). This is useful for automating the spread of tape volume selection across PVRs over time to ensure balanced usage of tapes.

Another way to change the priority is the custom method. In this method, the administrator will identify specific volumes and update their priority based on custom preferences. This could be used, for instance, to prioritize the usage of some specific tape volume for testing in a new tape library, or to prioritize using volumes which are in tape libraries with the most open drives at a given point in time.

For sites wanting to use the policy-based methods provided by HPSS, hpss_create_default_vv_select_weights(7) can be run for each storage class to generate a policy. The tool will generate a policy file with weights for each configured PVR based on the number of tape drives compatible with the given storage class definition. Additionally, example templates for configuring weights by PVR are available in `config/templates/vv_select_config_pvr.template`, and in `config/templates/vv_select_config_pvr_drive.template` for selection by PVR and drive type.

See the man pages for hpss_vv_select_priority(7), hpss_create_default_vv_select_weights(7), hpss_update_priority_by_pvr(7), and hpss_update_priority_by_pvr_drive(7) for more information.

19.1.2.3. Aspects of Selection Priority Weights

There are various aspects of determining the weights for volume selection priority that should be considered. The default configuration from hpss_create_default_vv_select_weights(7) will give each PVR a weight equal to the number of tape drives available for the storage class in that PVR. If the goal is optimal PVR usage this is likely to be a good starting point, but there may be other factors to consider that may mean the PVR should be favored or disfavored beyond the total tape drive count.

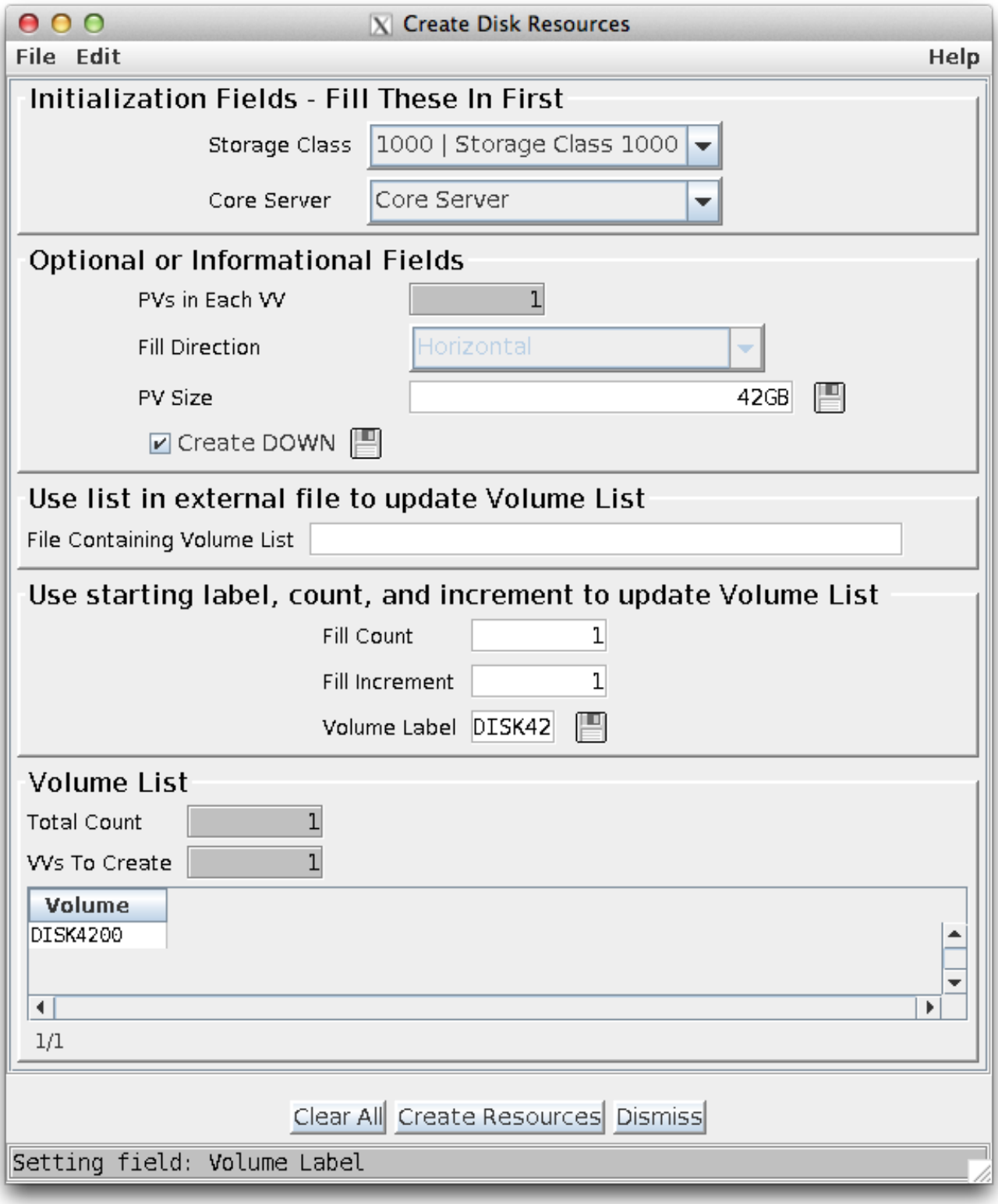
For example:

1. A tape library is new, and so it has a lesser read load compared to the other libraries. In this case, the weight may need to be increased, or the old tape library weights decreased, to cause HPSS to prefer the new tape library.
2. The tape library has a number of tape drives which are being used for ongoing internal operations such as tape verify operations. In this case, the weight may need to be decreased.
3. There may be a desire to balance tape library usage rather than to optimize the spread of data

across the tape libraries and configured tape drives (perhaps to split the data written by geography). In this case, the weights should be equalized, rather than weighted by tape drives. Set each PVR ID to 1 to round robin between PVRs.

4. There may be a temporary need to stop writing new tapes in a specific tape library. In this case, the weight should be set to 0.

19.1.2.4. Create Disk Resources window



This window is used to create disk storage resources, disk virtual volumes, in a Core Server. The

disks must first be imported to the appropriate PVL.

The names of the volumes may be entered into the window one at a time, or a list of volume names may be automatically generated from a single entry. Volume names are not entered directly into the **Volume List** at the bottom of the window but are typed into the **Volume Label** field.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List table:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
"AB7329"
```

The following shows an example of how the **Fill Direction** field works when set to `Horizontal` and

the **PVs in Each VV** value is "4" for the selected storage class.

Example:

```
PVs in Each VV = 4
Fill Direction = Horizontal
Fill Count = 8
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA007000","AA008000","AA009000","AA010000"
"AA011000","AA012000","AA013000","AA014000"
```

The following shows an example of how the **Fill Direction** field works when set to `Vertical` and the **PVs in Each VV** value is "4" for the selected storage class.

Example:

```
PVs in Each VV = 4
Fill Direction = Vertical
Fill Count = 8
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA007000","AA009000","AA011000","AA013000"
"AA008000","AA010000","AA012000","AA014000"
```

Once the Core Server completes the create request, SSM reports the number of physical volumes and virtual volumes created in the window status field.

Field descriptions

Initialization Fields - Fill These In First

Storage Class

The name of the storage class whose characteristics will be used to create the requested virtual volumes. The stripe width of this storage class will determine the number of columns in the **Volume List** table at the bottom of the window.

Core Server

The name of the Core Server that will process the create request and in which the disk storage resources will be created. The **Storage Class** field must be filled in before selecting a Core Server.

Optional or Informational Fields

PVs in Each VV

The stripe width of the selected storage class. This field may not be changed.

Fill Direction

Determines which direction filling takes place in the table. **Fill Direction** will not be enabled if the **PVs in Each VV** is "1". By default, if the **Fill Direction** field is enabled, it is set to `Horizontal`. Select the desired value for this field before entering a **Volume Label**.

PV Size

The size, in bytes, of each physical volume used. The default value is the value from the storage class. For disks, it should be changed on this window to the value that matches the actual value of each disk being imported.

Create DOWN

Creates the disk volumes in the DOWN state. DOWN disks will not be immediately usable and may be brought online by changing the volumes' VV Conditions to another state (RO, RW, or RWC). See *Core Server Disk Volume Information* window for information about a disk volume's VV Condition.

Use list in external file to update Volume List

File Containing Volume List

The name of an external file (accessible from the host on which the hpssgui(7) is executing) containing a list of volume labels to be added to the end of the **Volume List**.

Use starting label, count, and increment to update Volume List

Fill Count

The number of labels to be added to the **Volume List** when a value is typed into the **Volume Label** field. Valid values are positive integers greater than or equal to "1" up to the number of drives available. The default value is "1". If the list fills before the **Fill Count** is exhausted, filling stops and a message is displayed.

Fill Increment

This field determines how each new volume label is generated. The **Fill Increment** is the number by which each automatically generated label will differ from the previous one when a value is in the **Volume Label** field and the **Fill Count** is greater than "1". Valid values are positive integers up to the number of available drives. The default value is "1". A volume label is six alphanumeric characters. The **Fill Increment** is added to the least significant portion of a volume label to generate a new label.

Volume Label

The next label that will be generated and inserted into the table. The labels are generated when the field loses focus. Select **Fill Count**, **Fill Increment**, and **Fill Direction** before entering a value in this field.

Volume List

A list of volume labels specifying the volumes used to create resources. You cannot enter labels directly into this list but must instead use the **Fill Count**, **Fill Increment**, and **Volume Label**

fields. Use the scrollbar to navigate the list of labels.

Total Count

The total number of physical volumes in the generated Volume List. This is an informational field and not directly enterable.

VVs To Create

The number of virtual volumes that will be created. This is an informational field and not directly enterable.

Buttons

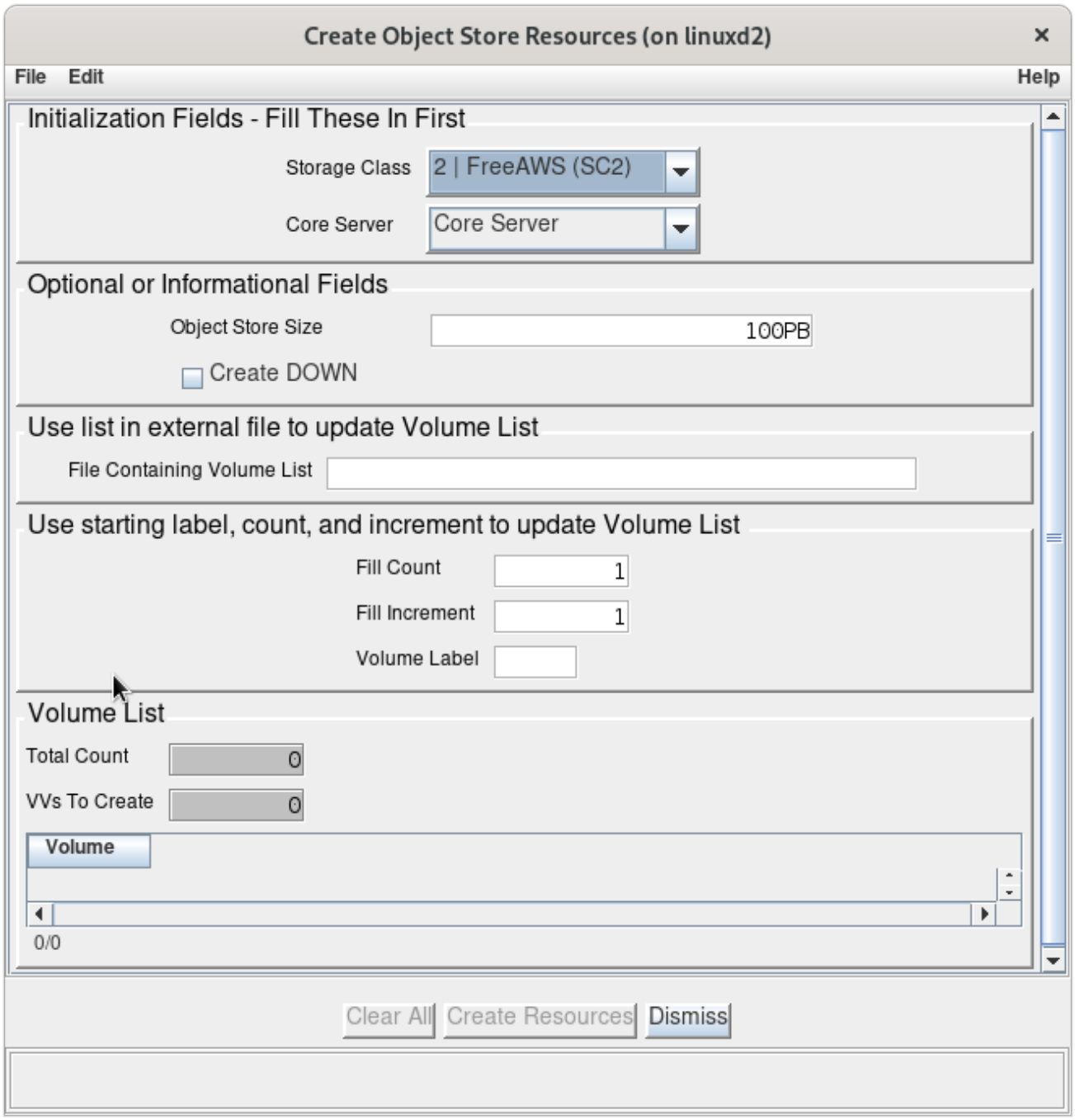
Clear All

This button clears the volume list and **Volume Label** field.

Create Resources

Begins the process of creating HPSS storage resources using the displayed data. A start message is displayed on the status bar, and all window features are disabled, except for the **Dismiss** button. This prevents the user from modifying any data on the window while the create request is in progress.

19.1.2.5. Create Object Store Resources window



This window is used to create object store storage resources, object store virtual volumes, in a Core Server. The object store drive must first be imported to the appropriate PVL.

The names of the volumes may be entered into the window one at a time, or a list of volume names may be automatically generated from a single entry. Volume names are not entered directly into the **Volume List** at the bottom of the window but are typed into the **Volume Label** field.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each

one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List table:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
"AB7329"
```

The following shows an example of how the **Fill Direction** field works when set to `Horizontal` and the **PVs in Each VV** value is "4" for the selected storage class.

Example:

```
PVs in Each VV = 4
Fill Direction = Horizontal
Fill Count = 8
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA007000","AA008000","AA009000","AA010000"
"AA011000","AA012000","AA013000","AA014000"
```

The following shows an example of how the **Fill Direction** field works when set to `Vertical` and the **PVs in Each VV** value is "4" for the selected storage class.

Example:

```
PVs in Each VV = 4
Fill Direction = Vertical
Fill Count = 8
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List table:
"AA007000","AA009000","AA011000","AA013000"
"AA008000","AA010000","AA012000","AA014000"
```

Once the Core Server completes the create request, SSM reports the number of physical volumes and virtual volumes created in the window status field.

Field descriptions

Initialization Fields - Fill These In First

Storage Class

The name of the storage class whose characteristics will be used to create the requested virtual volumes. The stripe width of this storage class will determine the number of columns in the **Volume List** table at the bottom of the window.

Core Server

The name of the Core Server that will process the create request and in which the object store storage resources will be created. The **Storage Class** field must be filled in before selecting a Core Server.

Optional or Informational Fields

Object Store Size

Informational limit to place on the number of bytes stored via this object storage resource. This value is used to drive threshold notifications based on the values configured for the associated storage class.



This setting will not limit the amount of data written to the object store. It is provided only as a tool to track the current vs. expected space usage.

Create DOWN

Creates the object store volumes in the DOWN state. DOWN object store volumes will not be immediately usable and may be brought online by changing the volumes' VV Conditions to another state (RO, or RWC). See *Core Server Object Store Volume Information* window for information about a object store volume's VV Condition.

Use list in external file to update Volume List

File Containing Volume List

The name of an external file (accessible from the host on which the hpssgui(7) is executing) containing a list of volume labels to be added to the end of the **Volume List**.

Use starting label, count, and increment to update Volume List

Fill Count

The number of labels to be added to the **Volume List** when a value is typed into the **Volume Label** field. Valid values are positive integers greater than or equal to "1" up to the number of drives available. The default value is "1". If the list fills before the **Fill Count** is exhausted, filling stops and a message is displayed.

Fill Increment

This field determines how each new volume label is generated. The **Fill Increment** is the number by which each automatically generated label will differ from the previous one when a value is in the **Volume Label** field and the **Fill Count** is greater than "1". Valid values are positive integers up to the number of available drives. The default value is "1". A volume label is six alphanumeric characters. The **Fill Increment** is added to the least significant portion of a volume label to generate a new label.

Volume Label

The next label that will be generated and inserted into the table. The labels are generated when the field loses focus. Select **Fill Count**, **Fill Increment**, and **Fill Direction** before entering a value in this field.

Volume List

A list of volume labels specifying the volumes used to create resources. You cannot enter labels directly into this list but must instead use the **Fill Count**, **Fill Increment**, and **Volume Label** fields. Use the scrollbar to navigate the list of labels.

Total Count

The total number of physical volumes in the generated Volume List. This is an informational field and not directly enterable.

VVs To Create

The number of virtual volumes that will be created. This is an informational field and not directly enterable.

Buttons

Clear All

This button clears the volume list and **Volume Label** field.

Create Resources

Begins the process of creating HPSS storage resources using the displayed data. A start message is displayed on the status bar, and all window features are disabled, except for the **Dismiss** button. This prevents the user from modifying any data on the window while the create request

is in progress.

19.2. Taking a tape library down for maintenance

When a library is down for maintenance, the administrator can mark its PVR as **disabled** in the PVR configuration. This will inform HPSS that the library is down and should not be used. While in this state, the SSM will show the PVR as disabled in the server window. This can be set whether the PVR is running or not.

While the PVR is disabled, HPSS will stop using that tape library for operations. This allows HPSS to gracefully handle library down scenarios.



The PVR can be marked disabled and enabled without a restart and the change takes effect immediately.

When the PVR is marked disabled, the following occur:

1. All import, mount, and dismount requests will be rejected.
2. All export requests which specify to eject the cartridge will be rejected.
3. Requests to read from tapes in the PVR will be rejected as if the tape were DOWN. This includes COS-specific read retry or stage retry behavior.
4. When the Core Server is looking for new tape volumes to begin writing to, it will skip this PVR.
5. Processing of any queued read or stage requests against a volume in this PVR will be paused.

It is still possible to move a cartridge from a disabled PVR to another PVR. In that scenario, the administrator should manually remove the volume from the down library and place it in the new library. This can be useful if a site needs immediate access to data on a cartridge in the disabled PVR.

Read and write operations against already mounted tape volumes will continue. These can be canceled by the administrator if desired. The volumes will not be dismounted by HPSS because the library has been marked as disabled. The PVL will periodically try to dismount these tapes until the PVR is brought back online. For short library outages, tapes can be left in the tape drives and when the library maintenance is over HPSS will pick up where it left off.



For outages that are expected to take hours, administrators may want to stop activity against the disabled PVR. It is recommended that the PVR be marked as disabled to avoid generating additional mount requests. Then the PVL jobs canceled. Then any remaining I/O jobs in flight canceled if needed. The administrator can dismount the drives in the disabled PVR using **umccp** (the **dismount** command) or similar tools if the library is functionally able to complete those operations. When the administrator dismounts tapes outside of HPSS in this situation, the PVR will issue an alarm that it could not dismount the drive once it has been repaired, but there will be no effect on operations.

The disabled state of the volumes is shown in dump_sspvs(7). The disabled flag of the PVR is also available in lshpss(7) and dumppv_pvr(7).

When the tape library becomes usable again, if there were no changes to the library configuration (e.g. number of tape drives, number of storage slots), then the disabled flag can be cleared or the PVR marked repaired to make the PVR immediately available for use by the system again. Otherwise, the PVR can be restarted to pick up any changes, and then repaired.

19.3. Removing storage space

Removing storage space from the HPSS system is a two-step process:

1. **Delete Resources**. In the first step, the Core Server virtual volume metadata is deleted; this process is called deleting storage resources.
2. **Export Volumes**. In the second step, the physical volumes that belonged to the virtual volumes are exported from the PVL.

The second step, exporting the volumes from the PVL, is not always necessary. Once the resources are deleted in Step 1, new storage resources can be created on the volumes without the need to export the volumes from the system and reimport them. Exporting the volumes is necessary only if the volumes are not going to be reused in the HPSS system, or if for some reason they must be relabeled. The site may wish to export the volumes if they are to be imported into a different PVR in the system, but this function can also be accomplished with the Move Cartridges operation.

19.3.1. Deleting storage resources

Storage resources can be removed from HPSS volumes by the use of the *Delete Resources* window or the remove(7) utility.

Once resources have been deleted, the volumes may be reused or removed from the HPSS system. To reuse the volumes, create new storage resources on them using the *Create Resources* window. To remove them entirely from the HPSS system, export them from the PVL using the *Export Volumes* window.

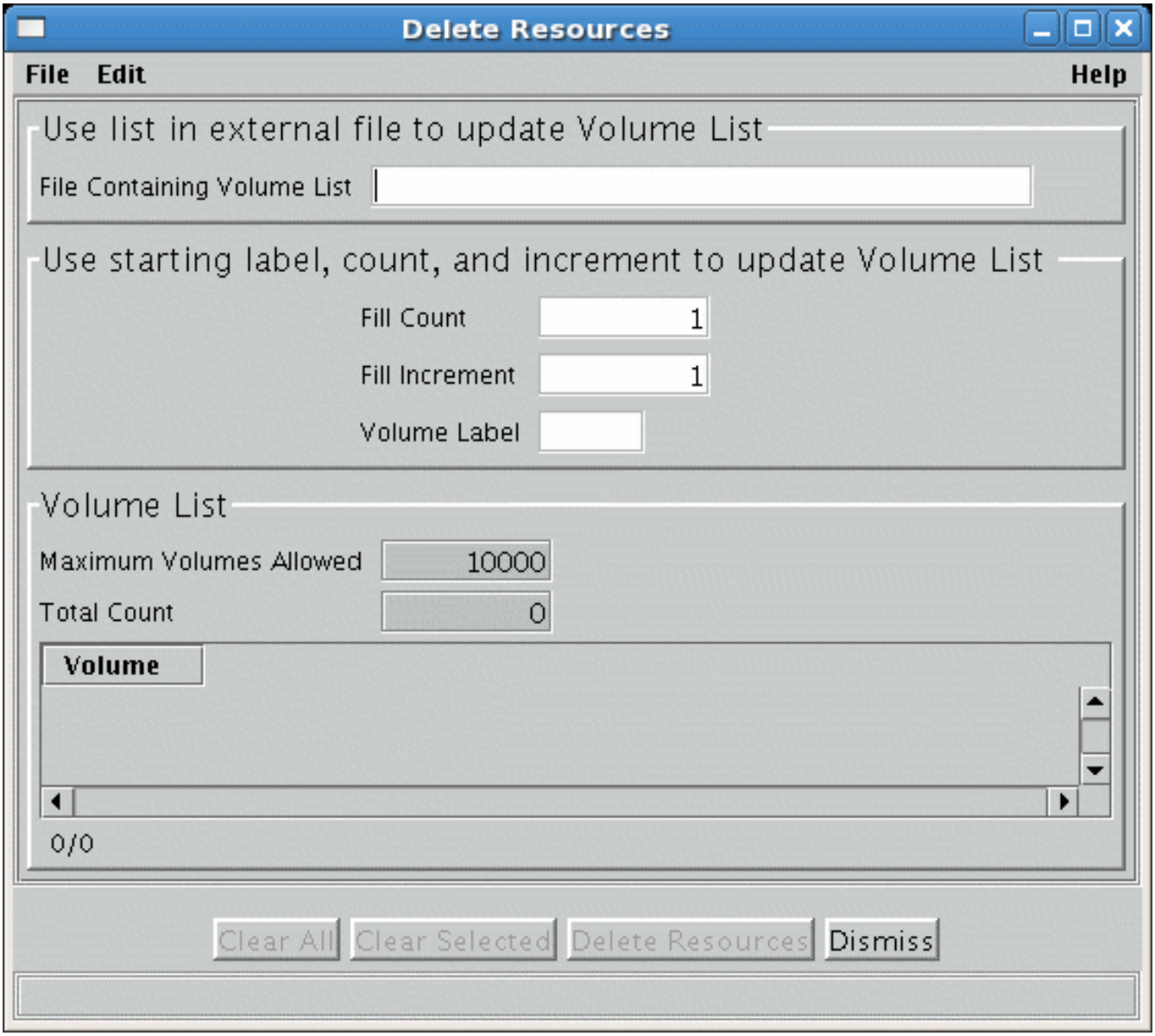
19.3.1.1. Rules for deleting resources

Volumes on which resources are to be deleted must be empty. There must be no storage segments associated with the volumes. The *Core Server Disk Volume* or *Core Server Tape Volume* window must show a zero in the **Number of Active Segments** field, and the VV Condition must be EMPTY. If any storage segments remain on a volume, the resource deletion operation will fail. You can use the volume repack(7) features of HPSS to empty disk, tape, and object store volumes so that their resources may be deleted.

The Delete Resources operation may be performed on either retired or non-retired volumes.

Successful deletions result in the removal of the Core Server's metadata records from the appropriate subsystem. The volume is then no longer usable for storage by HPSS, but it is still recognized by the PVL.

19.3.1.2. Delete Resources window



This window is used to delete empty or unused storage resources (that is, virtual volumes) from a Core Server. The volumes must have no storage segments on them when they are removed.

The list of the volumes to be deleted may be constructed in any of three ways. Each volume name may be typed in one at a time, or a list of volume names may be automatically generated from a single entry, or a list of volume names may be specified from an input file. Volume names are not typed directly into the **Volume List** at the bottom of the window.

Any of the three entry methods may be repeated multiple times on the same window to add additional volumes to the list. All three entry methods or any combination of them may be used in succession on the same window.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the

Volume Label field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
"AB7329"
```

The filling will not occur and an error will be displayed if the specified values would generate an invalid volume label (for example, one greater than zzz999).

To specify a list of volumes from a file, create a file containing the name of each volume to be deleted on a separate line. Volume names must be six alphanumeric characters. No other characters are allowed on the line. The file must be accessible from the host on which the hpssgui is executing. Enter the name of the file in the **File Containing Volume List** field. The volume names from the file will be added to the **Volume List**.

Field descriptions

File Containing Volume List

The name of an external file containing a list of volume labels to be added to the end of the **Volume List**.

Fill Count

The number of volume labels to be added to the end of the **Volume List** when the **Volume Label** field is next modified.

Fill Increment

In a multiple-label fill, this field determines how each new PV label is generated from the previous one. Each new volume label entered in the table is the previous entry incremented by this amount.

Volume Label

The label to be added to the end of the PV list. If **Fill Count** is greater than "1", this will be the first label of a fill sequence. The label must be exactly six characters long. Entering a value in this field causes the table entries to be generated.

Maximum Volumes Allowed

The maximum number of volume labels that will fit in the **Volume List**. This field is informational and not enterable.

Total Count

The total number of volumes in the list to be deleted. This is the current size of the generated **Volume List**.

Volume List

A list of volume labels of volumes to be deleted. You cannot enter labels directly into this list but must construct it using one of the three methods described above. You can select one or more labels from the list for the purpose of removing them from the list (see **Clear Selected** below). To select one label, click on it with the mouse; the selection will be highlighted. To select a range, select one label; then select a second label while holding the Shift key. The selected labels, and all labels between them, will be highlighted. You may also hold down the Control key while selecting, to select or deselect individual labels without affecting any other selections.

Buttons

Clear All

Clears the **Volume List** and resets **Fill Count** and **Fill Increment**.

Clear Selected

If one or more volumes are selected (highlighted) in the **Volume List**, clicking on this button will remove them from the list. Note that this does not delete anything from HPSS.

Delete Resources

Begins the deletion of resources using the displayed data. A start message is displayed on the status bar, and all window features are disabled except for the **Dismiss** button. This prevents the user from modifying any data on the window while the delete request is in progress.

Related information

Dealing with a space shortage

19.3.2. Exporting volumes from HPSS

Exporting volumes from HPSS is the process of removing tape cartridges, disk volumes, or object store volumes from HPSS control.

19.3.2.1. Rules for exporting volumes

Tape cartridges may be physically exported from any managing robotic library. To export a tape cartridge from HPSS, the administrator must be familiar with the operation of the PVR from which the cartridge will be removed because the physical cartridge ejection process differs among the PVRs supported by HPSS:

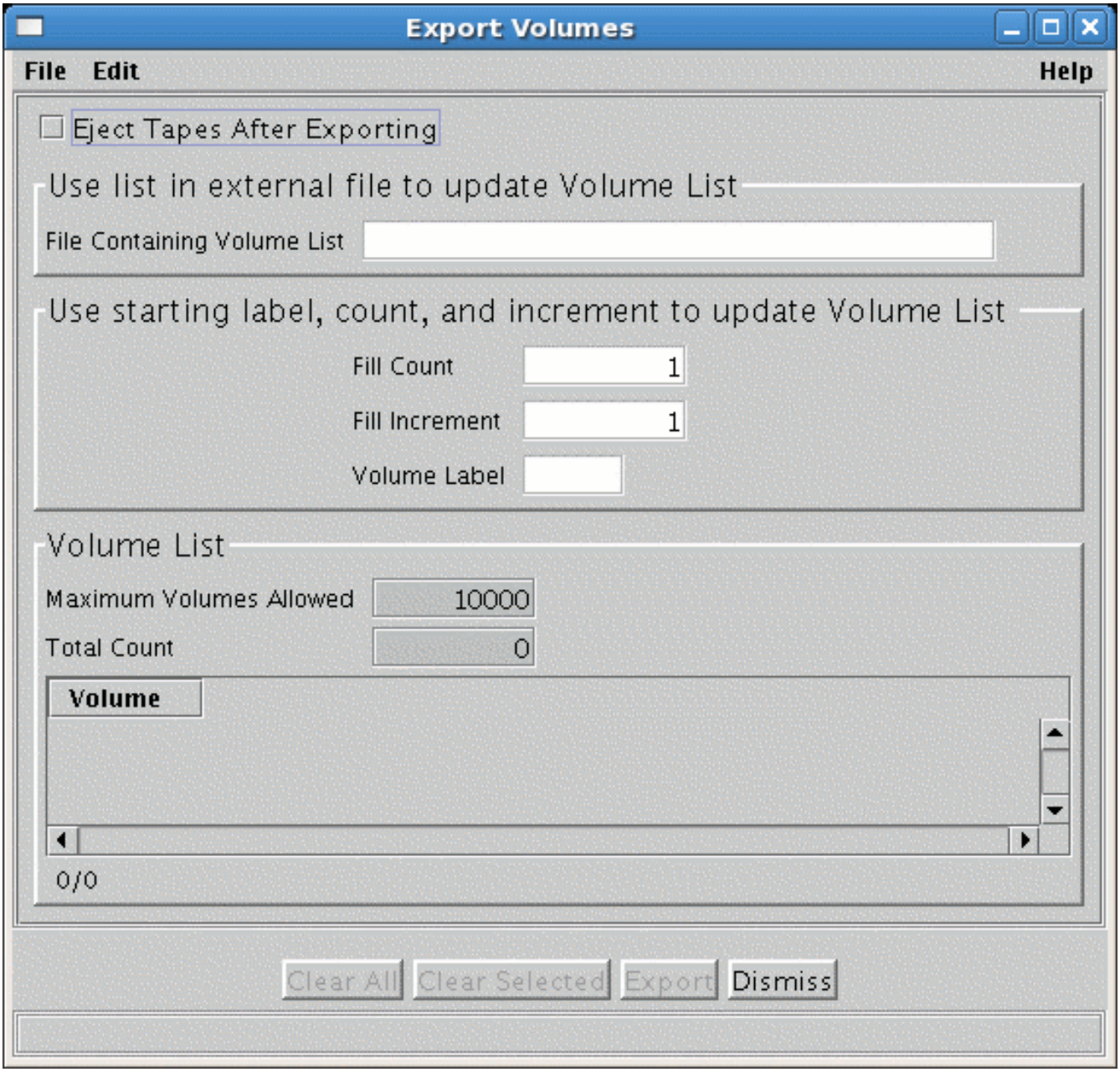
- **Operator** - No cartridge ejection step is necessary (or possible).
- **SCSI** - When an HPSS tape export command is issued, the cartridges will be placed in the input/output (I/O) slot and can then be manually removed.
- **STK** - The exported cartridges will be placed in the CAP and can then be manually removed. The export operation is *not* completed until cartridges are manually removed.

Only imported but unallocated volumes can be exported. Volumes allocated to a Core Server can be unallocated by deleting all Core Server metadata that describe the volume using the procedure described in Deleting storage resources.

When exporting a removable cartridge from a robot, the cartridges will normally be sent to the convenience I/O port if it is available (this will not be done if the Export Without Eject option is used). If no locations are available for the cartridge to be ejected, an alarm will appear on the *Alarms and Events* window and HPSS will periodically retry the eject operation.

Use the *Export Volumes* window in the following section to perform the actual export operation.

19.3.2.2. Export Volumes window



This window allows you to export tape, disk, and object store volumes from the HPSS system. Exporting a volume is equivalent to telling HPSS that the volume no longer exists. Before volumes can be exported, the Core Server storage resources that describe the volumes must be deleted using the procedure described in Deleting storage resources.

The list of the volumes to be exported may be constructed in any of three ways. Each volume name may be typed in one at a time, or a list of volume names may be automatically generated from a single entry, or a list of volume names may be specified from an input file. Volume names are not entered directly into the **Volume List** at the bottom of the window.

Any of the three entry methods may be repeated multiple times on the same window to add additional volumes to the list. All three entry methods or any combination of them may be used in succession on the same window.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** fields each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
"AB7329"
```

The filling will not occur and an error will be displayed if the specified values would generate an invalid volume label (for example, one greater than zzz999).

To specify a list of volumes from a file, create a file containing the name of each volume to be exported on a separate line. Volume names must be six alphanumeric characters. No other characters are allowed on the line. The file must be accessible from the host on which the hpssgui(7) is executing. Enter the name of the file in the **File Containing Volume List** field. The volume names from the file will be added to the **Volume List**.

The entire list of volumes is sent to the System Manager as a single export request. The System Manager processes them one at a time, and if it encounters an error, it stops and returns. In this

case, the window status line will report the number of successful exports, and the volume name that caused the failure. Since the System Manager does not ignore volumes that have already been exported, but instead considers them failed exports, you cannot retry an export by resubmitting the same list of volumes. You will first have to remove the successfully exported volumes from the list. The **Clear Selected** button can be useful for this.

Once the export is completed, cartridges residing in robotic tape systems will be ejected by the controlling PVR (if **Eject Tapes after Exporting** is selected) and should be removed from the access port of the robot.

Field descriptions

Eject Tapes After Exporting

If this check box is selected, the exported tape volumes will also be ejected from the PVR.

File Containing Volume List

The name of an external file containing a list of volume labels to be added to the end of the **Volume List**.

Fill Count

The number of volume labels to be added to the end of the list when the **Volume Label** field is next modified. This number may be one or greater. If the list fills up before the **Fill Count** is exhausted, filling stops, and a message box is displayed (see **Maximum Volumes Allowed** below).

Fill Increment

In a multiple-label fill, this field determines how each new volume label is generated from the previous one. A volume label is six alphanumeric characters. The **Fill Increment** is added to the numerical part of a volume label to generate a new label.

Volume Label

The volume label to be added to the end of the **Volume List**. If **Fill Count** is greater than "1", this will be the first label of a fill sequence. The label must be exactly six characters long. Lower case characters entered into this field will be converted to upper case.

Maximum Volumes Allowed

The maximum number of volume labels that will fit in the **Volume List**. This value is 10,000 and is maintained by SSM. This field is non-editable.

Total Count

The total number of volumes currently in the **Volume List**.

Volume List

A list of volume labels specifying the volumes to be exported. You cannot enter labels directly into this list, but must construct it using one of the three ways described above. Use the scrollbar to move up and down the list of labels.

You can select one or more labels from the list for the purpose of deleting them from the list (see

Clear Selected below).

Buttons

Clear All

Clears the **Volume List** and resets **Fill Count** and **Fill Increment** to "1".

Clear Selected

If one or more volumes are selected (highlighted) in the **Volume List**, clicking on this button will remove them from the list. Note that this does not actually delete anything from HPSS.

Export

Begins the volume export using the displayed data. A start message is displayed on the status bar at the bottom of the window, and all window features are disabled, except for the **Dismiss** button; this prevents you from modifying any data on the window while the export is in progress.

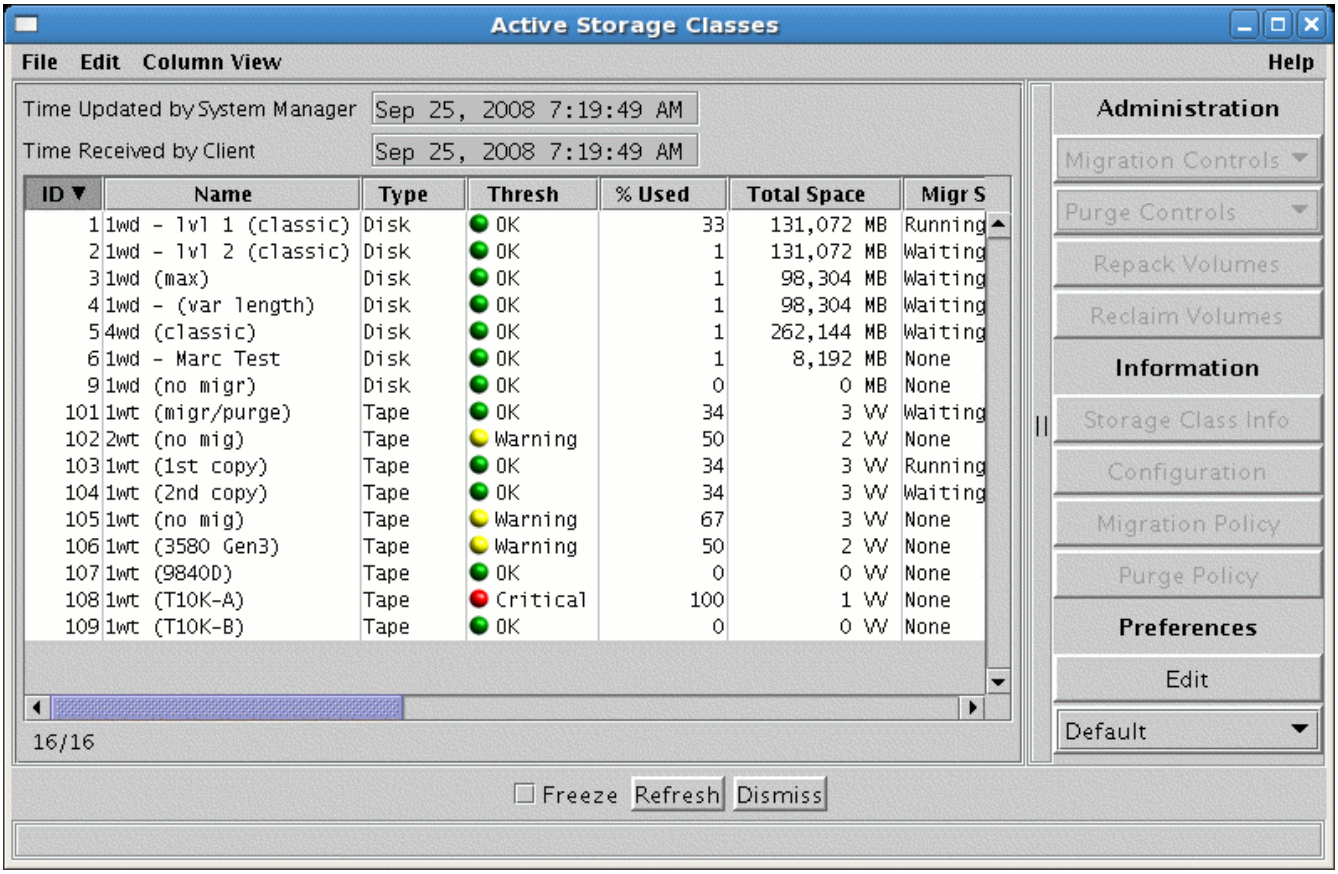
19.4. Monitoring storage space

One of the most important administrative tasks is to monitor storage space usage in the HPSS system. The goal is early detection of any potential space shortages so that corrective action can be taken in a timely manner. Each storage class is configured with a warning threshold and a critical threshold to inform SSM users when the free space in a storage class runs low. Warning and major alarms are sent to the *Alarms and Events* window periodically when the warning and critical thresholds are exceeded. Migration policies can be configured to automatically start migration when either threshold is exceeded.

When a storage class experiences a threshold-exceeded condition, an administrator may need to review the associated migration and purge policies for the storage class as well as the total storage space available to the storage class. The migration and purge policies may need to be modified to free up more space or to free up the space more frequently. In addition, the total storage space for the storage class may need to be reviewed to determine whether it is sufficient to accommodate the actual usage of the storage class.

Storage space in HPSS is monitored from the *Active Storage Classes* window.

19.4.1. Active Storage Classes window



The *Active Storage Classes* window allows you to view a list of all active storage classes. An active storage class is defined as a storage class configured in HPSS to which storage volumes have been assigned. Volumes are assigned to a storage class by the Create Resources operation.

The information displayed in this window is obtained from the MPS, which obtains part of it from the Core Servers.



The Migration/Purge Server must be running and connected to SSM or the Active Storage Classes window will be stale. All Core Servers must be running or the storage classes managed by them will not be reported in this window.

If multiple storage subsystems exist, this window will display a list of storage classes for each subsystem. To display this window, from the *HPSS Health and Status* window, select the **Monitor** menu, and from that select the **Storage Classes Active** menu item.

The function buttons to the right of this list require that one storage class row be selected from the list. To select a row, click on it with the mouse; the selection will be highlighted. Note that when you select a row, you are selecting a storage class within a particular storage subsystem.

See also the related window Configured Storage Classes window.

The *Configured Storage Classes* window lists all configured storage classes in the system, whether or not any storage resources have been assigned to them.

Field descriptions

Active Storage Classes list

The **Column View** menu item may be used to determine which of the following columns are displayed in the list:

Subsystem

The name of the storage subsystem to which the storage class information applies. It is possible to have multiple rows in the table with identical storage class ID and Name columns, each for a different storage subsystem.

ID

The unique storage class ID number defined when the storage class is configured.

Name

The name of the storage class.

Type

The storage class type. Possible values are `Disk`, `Tape` or `Object Store`.

Thresh

The current threshold status for the storage class. For disk and object store, a threshold is defined as a percentage of the total bytes of storage space. When the percentage of storage space used rises above a threshold level, the threshold is exceeded. For tape, a threshold is defined as a count of free virtual volumes. When the number of free volumes drops below a threshold level, the threshold is exceeded. Possible values are:

- `OK` indicates the storage class has not reached its warning threshold.
- `Stale` indicates the storage class is active but the MPS responsible for it is shutdown, has died, or is otherwise inaccessible to SSM.
- `Warning` indicates the storage class has reached its warning threshold, but has not reached its critical threshold.
- `Critical` indicates the storage class has reached its critical threshold.

Free Space

The amount of free space left in the storage class. For disk and object store, this is the number of free bytes. For tape, it is the number of free virtual volumes.

% Free

The **Free Space** value expressed as a percentage of **Total Space**.

Used Space

The amount of used space in the storage class. For disk and object store, this is the number of used bytes. For tape, it is the number of used virtual volumes.

% Used

The **Used Space** value expressed as a percentage of **Total Space**.

Partially Filled VVs

The number of tapes that have been written, but have not yet reached EOM. This field will be blank for disk and object store storage classes.

Total Space

The total amount of space in the storage class. For disk and object store, this is the number of bytes. For tape, it is the number of virtual volumes.

Migr State

The migration state for the storage class. The **Migration State** is preserved when the MPS is recycled. The exception is `Running` which will be set to `Waiting` upon MPS recycle. Possible values are:

- `Waiting` indicates migration is not taking place at this time. The start of the next migration is waiting until criteria specified in the migration policy are met.
- `Running` indicates a migration is in progress.
- `None` indicates the storage class, as configured, is not a candidate for migration.
- `Suspended` indicates migration has been suspended.
- `Disabled` indicates migration has been stopped.

Migr Policy

The migration policy assigned to the storage class.

Purge State

The purge state for the storage class. The **Purge State** is preserved when the MPS is recycled. The exception is `Running` which will be set to `Waiting` upon MPS recycle. Possible values are:

- `Waiting` indicates purging is not taking place at this time. The start of the next purge is waiting until criteria specified in the purge policy are met.
- `Running` indicates a purge is in progress.
- `None` indicates the storage class, as configured, is not a candidate for purging.
- `Suspended` indicates purging has been suspended.
- `Disabled` indicates purging has been stopped.

Purge Policy

The purge policy assigned to the storage class.

Administration buttons

Migration Controls

This drop-down list can be used to send migration commands to the MPS on the selected storage classes. This drop-down list is enabled only if at least one of the selected storage classes has been assigned a migration policy. Click on it to drop down a list of control options. These are:

`Start`

If the migration state is `Waiting`, this starts the migration and puts it into the `Running` state.

This does not guarantee that any files will be migrated, only that files will be reevaluated against the appropriate migration policy and migrated as appropriate.

`Stop`

If the migration state is `Running`, this stops the migration and puts it into the `Disabled` state. Any ongoing I/O will be aborted.

`Suspend`

If the migration state is `Waiting` or `Running`, this puts the migration into the `Suspended` state. Any ongoing I/O will be aborted.

`Resume`

If the migration state is `Suspended`, this returns it to `Waiting` and allows MPS to again begin scheduling migration runs.

`Reread policy`

Tells the MPS to refresh its migration policy information by rereading the policy.



These controls are not enabled for object store storage classes.

Purge Controls

This drop-down list can be used to send purge commands to the MPS on the selected storage classes. This drop-down list is enabled only if at least one of the selected storage classes has been assigned a purge policy. Click on it to drop down a list of control options. These are:

`Start`

If the purge state is `Waiting`, this starts the purge and puts it into the `Running` state. This does not guarantee that any files will be purged, only that files will be reevaluated against the appropriate purge policy and purged as appropriate.

`Stop`

If the purge state is `Running`, this stops the purge and puts it into the `Disabled` state.

`Suspend`

If the purge state is `Waiting` or `Running`, this puts the purge into the `Suspended` state.

`Resume`

If the purge state is `Suspended`, this returns it to `Waiting` and allows MPS to again begin scheduling purge runs.

`Reread policy`

Tells the MPS to refresh its purge policy information by rereading the policy.



These controls are not enabled for object store storage classes.

Repack Volumes

Opens the *Repack Virtual Volumes* window, allowing you to start the repack(7) utility program on the selected storage class. See Repack Virtual Volumes window.

for details. This button is only enabled for tape storage classes.

Reclaim Volumes

Opens the *Reclaim Virtual Volumes* window, allowing you to start the reclaim(7) utility program on the selected storage class. See Reclaim Virtual Volumes window.

for details. This button is only enabled for tape storage classes.

Information buttons

Storage Class Info

Opens the *MPS Storage Class Information* window for the selected storage class. For detailed information on usage of this window, see MPS Disk Storage Class Information.

Configuration

Opens the *Storage Class Configuration* window for the selected storage class.

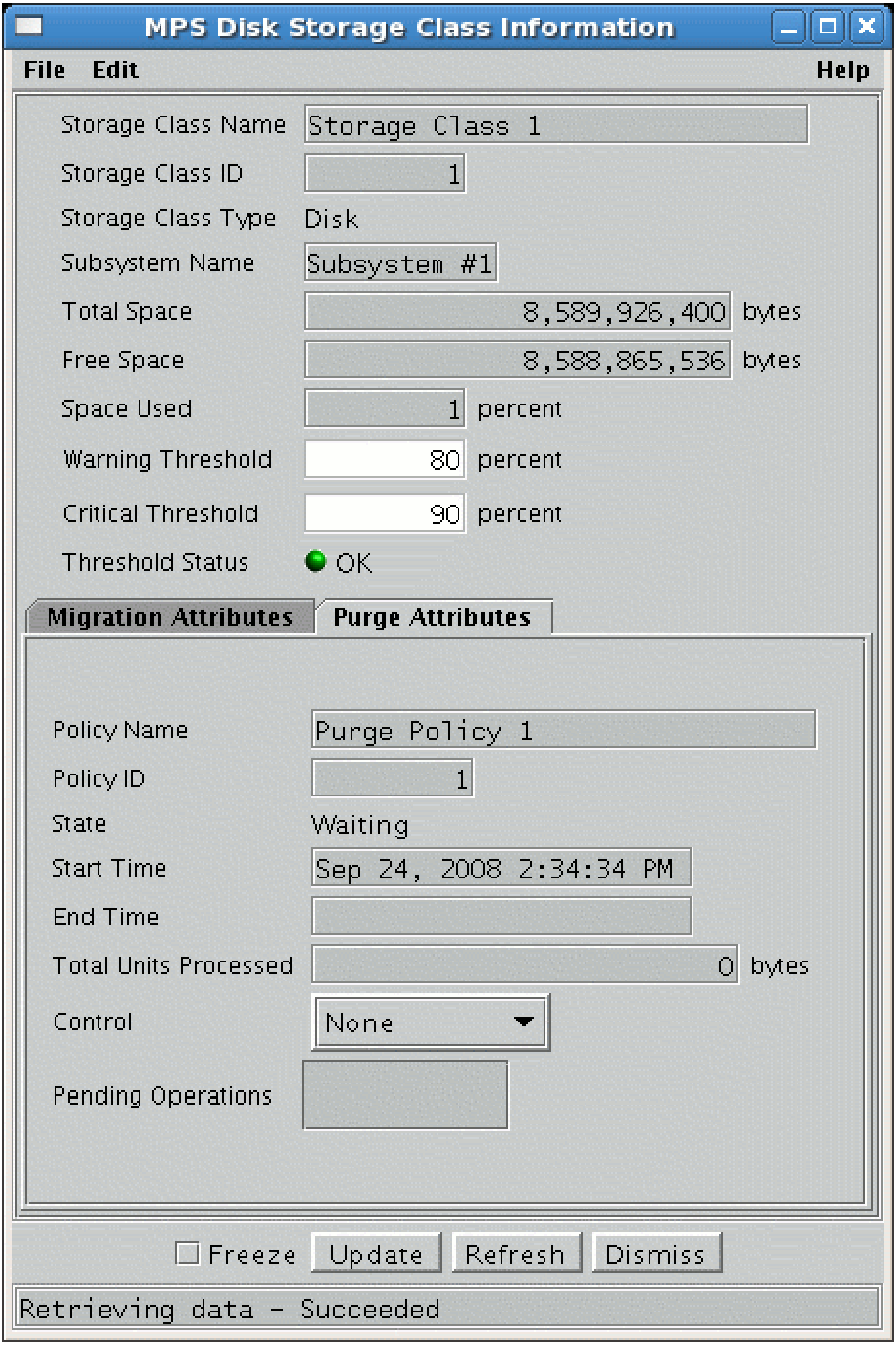
Migration Policy

Opens the *Migration Policy* window for the migration policy assigned to the selected storage class. If there is no assigned migration policy, the button is disabled.

Purge Policy

Opens the *Purge Policy* window for the purge policy assigned to the selected storage class. If there is no assigned purge policy, the button is disabled.

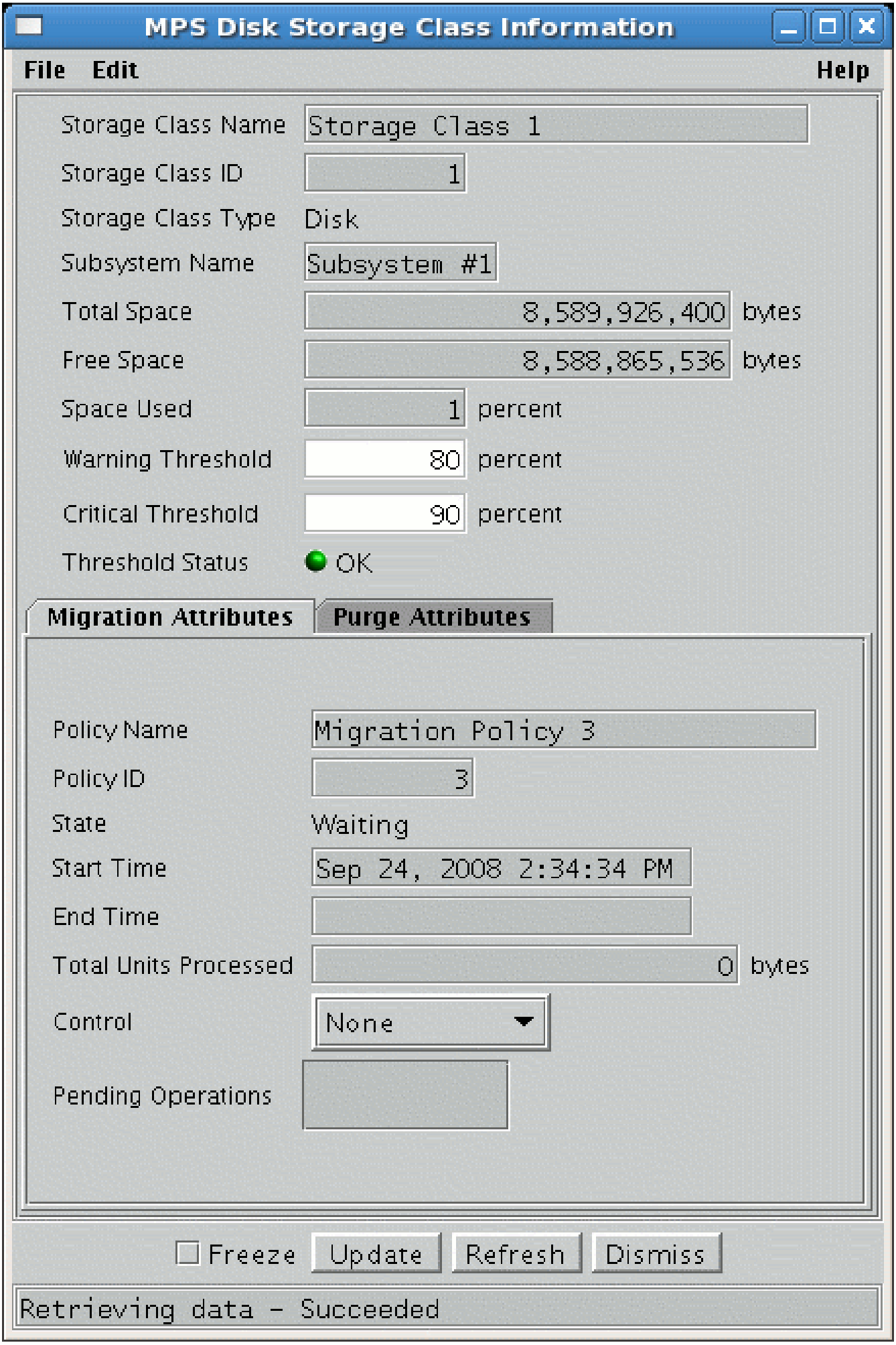
19.4.2. MPS Disk Storage Class Information


MPS Disk Storage Class Information
File
Edit
Help
Storage Class Name
Storage Class 1
Storage Class ID
1
Storage Class Type
Disk
Subsystem Name
Subsystem #1
Total Space
8,589,926,400
bytes
Free Space
8,588,865,536
bytes
Space Used
1
percent
Warning Threshold
80
percent
Critical Threshold
90
percent
Threshold Status
OK
Migration Attributes
Purge Attributes
Policy Name
Purge Policy 1
Policy ID
1
State
Waiting
Start Time
Sep 24, 2008 2:34:34 PM
End Time
Total Units Processed
0
bytes
Control
None
Pending Operations
Freeze
Update
Refresh
Dismiss
Retrieving data - Succeeded

This window allows you to view and update the information associated with an active disk storage class. It reports the storage space data as well as any exceeded thresholds. The window also reports detailed information on the migration and purge status. In addition, the window allows the SSM user to control the migration and purge process to override the associated migration and purge policies.

There are three differences between the disk and tape storage class information windows. First, the disk window reports space in terms of bytes while the tape window reports space in terms of virtual volumes (VVs). Second, the disk window reports *space* used while the tape window reports *percent* used. Third, the disk window shows purge status while the tape window does not.

Since an MPS is specific to a storage subsystem, the information in this window applies only to the volumes of the storage class which are assigned to a single storage subsystem. It is possible for multiple subsystems to use a storage class. If two of these windows are opened, each for the same storage class but for different subsystems, they will display different information.


MPS Disk Storage Class Information
File
Edit
Help
Storage Class Name
Storage Class 1
Storage Class ID
1
Storage Class Type
Disk
Subsystem Name
Subsystem #1
Total Space
8,589,926,400
bytes
Free Space
8,588,865,536
bytes
Space Used
1
percent
Warning Threshold
80
percent
Critical Threshold
90
percent
Threshold Status
OK
Migration Attributes
Purge Attributes
Policy Name
Migration Policy 3
Policy ID
3
State
Waiting
Start Time
Sep 24, 2008 2:34:34 PM
End Time
Total Units Processed
0
bytes
Control
None
Pending Operations
Freeze
Update
Refresh
Dismiss
Retrieving data - Succeeded

Any changes made to fields on the window are sent directly to the MPS and are effective immediately. If the MPS is restarted, changes made through this window are lost.

Field descriptions

Storage Class Name

The name assigned to this storage class.

Storage Class ID

The numeric ID of this storage class.

Storage Class Type

The class of media assigned to this storage class (tape, disk or object store).

Subsystem Name

The name of the storage subsystem for which the storage class information is being displayed.

Total Space

For disk storage classes this reports the total capacity of the storage class in bytes.

Free Space

For disk storage classes this reports the unused capacity of the storage class in bytes.

Space Used

The percent of storage space in the storage class which is used. This is calculated from **Total Space** and **Free Space** and reported as a percentage.

Warning Threshold

For disk storage classes, this value is a percentage of total storage space. When the used space in this storage class exceeds this percent of the total space, the MPS will send a warning alarm to SSM.

Critical Threshold

For disk storage classes, this value is a percentage of total storage space. When the used space in this storage class exceeds this percent of the total space, the MPS will send a critical alarm to SSM.



Note that the disk threshold reaches the warning level when the percent of used space rises above the threshold value, while the tape threshold reaches the warning level when the free volume count drops below the threshold value.

If you wish to disable the threshold so it will not generate any alarms, set this field to 100 percent for disk, and "0" volumes for tape.

Threshold Status

The current used space threshold status. Possible values are `OK`, `Warning`, and `Critical`.

Migration Attributes tab

This panel contains the current migration status for this storage class.

Policy Name

The name of the migration policy assigned to this storage class.

Policy ID

The ID of the migration policy assigned to this storage class.

State

The current migration state for this storage class. Possible values are:

- `None` indicates this storage class, as configured, is not a candidate for migration.
- `Running` indicates a migration on this storage class is in progress.
- `Waiting` indicates migration is not taking place at this time. The start of the next migration is waiting until criteria specified in the migration policy are met.
- `Suspended` indicates migration on this storage class has been suspended by the system administrator.
- `Disabled` indicates migration on this storage class has been stopped by the system administrator.

Start Time

The date and time when the most recent migration run started. It may still be running.

End Time

The date and time when the last migration run completed.

Total Units Processed

The amount of space in the storage class which has been migrated during the current or most recent migration run. For disk storage classes and for tape storage classes running the tape file migration algorithm, this is a number of bytes.

Control

This drop-down list can be used to send commands to the MPS regarding migration on this storage class. Click on it to drop down a list of control options. These are:

`None`

Nothing has been selected for the migration control.

`Start`

If the migration state is Waiting, this starts the migration and puts it into the Running state.

`Stop`

If the migration state is Running, this stops the migration and puts it into the Disabled state.

`Suspend`

If the migration state is Waiting or Running, this puts the migration into the Suspended state.

`Resume`

If the migration state is Suspended, this returns it to Waiting and allows MPS to again begin scheduling migration runs.

`Reread policy`

Tells the MPS to refresh its migration policy information by rereading the policy.

Pending Operations

When the MPS cannot respond immediately to a control command, a command may be saved as pending. Any such pending operations are displayed here.

Purge Attributes tab

This panel contains the current purge status for this storage class. This tab appears only on the *MPS Disk Storage Class Information* window.

Policy Name

The name of the purge policy assigned to this storage class.

Policy ID

The ID of the purge policy assigned to this storage class.

State

The current purge state for this storage class. Possible values are:

- `None` indicates this storage class, as configured, is not a candidate for purging.
- `Running` indicates a purge on this storage class is in progress.
- `Waiting` indicates purging is not taking place at this time. The start of the next purge is waiting until criteria specified in the purge policy are met.
- `Suspended` indicates purging on this storage class has been suspended.
- `Disabled` indicates purging on this storage class has been stopped.

Start Time

The date and time when the most recent purge run started. It may still be running.

End Time

The date and time when the last purge run completed.

Total Units Processed

The amount of space in the storage class which has been purged, in bytes.

Control

This button can be used to send commands to the MPS regarding purging on this storage class. Click on the button to drop down a list of control options. These are:

`None`

Nothing has been selected for the purge control.

`Start`

If the purge state is `Waiting`, this starts the purge and puts it into the `Running` state.

`Stop`

If the purge state is `Running`, this stops the purge and puts it into the `Disabled` state.

`Suspend`

If the purge state is `Waiting` or `Running`, this puts the purge into the `Suspended` state.

`Resume`

If the purge state is `Suspended`, this returns it to `Waiting` and allows MPS to again begin scheduling purge runs.

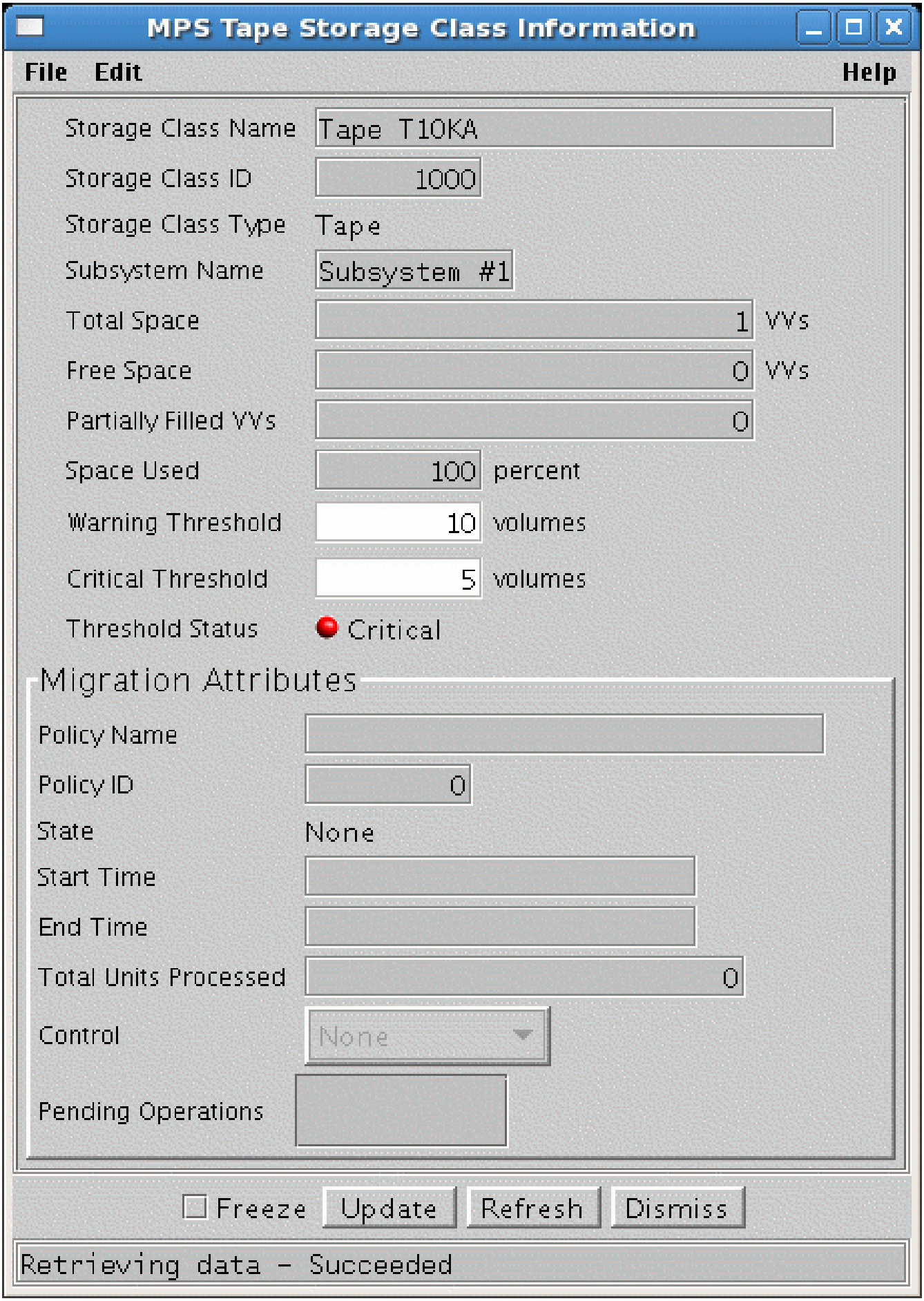
`Reread policy`

Tells the MPS to refresh its purge policy information by rereading the policy.

Pending Operations

When the MPS cannot respond immediately to a control command, a command may be saved as pending. Any such pending operations are displayed here.

19.4.3. MPS Tape Storage Class Information



This window allows you to view and update the information associated with an active tape storage class. It reports the storage space data as well as any exceeded thresholds. The window also reports

detailed information on the migration status. In addition, the window allows the SSM user to control the migration process to override the associated migration policies.

There are three differences between the disk and tape storage class information windows. First, the disk window reports space in terms of bytes while the tape window reports space in terms of virtual volumes (VVs). Second, the disk window reports *space* used while the tape window reports *percent* used. Third, the disk window shows purge status while the tape window does not.

Since an MPS is specific to a storage subsystem, the information in this window applies only to the volumes of the storage class which are assigned to a single storage subsystem. It is possible for multiple subsystems to use a storage class. If two of these windows are opened, each for the same storage class but for different subsystems, they will display different information.

Any changes made to fields on the window are sent directly to the MPS and are effective immediately.

Field descriptions

Storage Class Name

The name assigned to this storage class.

Storage Class ID

The numeric ID of this storage class.

Storage Class Type

The class of media assigned to this storage class (tape, disk or object store).

Subsystem Name

The name of the storage subsystem for which the storage class information is being displayed.

Total Space

For tape storage classes this reports the total capacity of the storage class in virtual volumes (VVs).

Free Space

For tape storage classes this reports the number of unwritten virtual volumes (VVs) in the storage class.

Partially Filled VVs

The number of tapes that have been written, but have not yet reached EOM.

Space Used

The percent of storage space in the storage class which is used. This is calculated from **Total Space** and **Free Space** and reported as a percentage.

Warning Threshold

For tape storage classes, this value is a count of free tape volumes. When the number of free volumes in this storage class drops below this level, the MPS will send a warning alarm to SSM.

Critical Threshold

For tape storage classes, this value is a count of free tape volumes. When the number of free volumes in this storage class drops below this level, the MPS will send a critical alarm to SSM.



Note that the disk threshold reaches the warning level when the percent of used space rises above the threshold value, while the tape threshold reaches the warning level when the free volume count drops below the threshold value.

If you wish to disable the threshold so it will not generate any alarms, set this field to 100 percent for disk, and "0" volumes for tape.

Threshold Status

The current used space threshold status. Possible values are OK, Warning, and Critical.

Migration Attributes

This panel contains the current migration status for this storage class.

Policy Name

The name of the migration policy assigned to this storage class.

Policy ID

The ID of the migration policy assigned to this storage class.

State

The current migration state for this storage class. Possible values are:

- `None` indicates this storage class, as configured, is not a candidate for migration.
- `Running` indicates a migration on this storage class is in progress.
- `Waiting` indicates migration is not taking place at this time. The start of the next migration is waiting until criteria specified in the migration policy are met.
- `Suspended` indicates migration on this storage class has been suspended by the system administrator.
- `Disabled` indicates migration on this storage class has been stopped by the system administrator.

Start Time

The date and time when the most recent migration run started. It may still be running.

End Time

The date and time when the last migration run completed.

Total Units Processed

The amount of space in the storage class which has been migrated during the current or most recent migration run.

Control

This drop-down list can be used to send commands to the MPS regarding migration on this

storage class. Click on it to drop down a list of control options. These are:

`None`

Nothing has been selected for the migration control.

`Start`

If the migration state is Waiting, this starts the migration and puts it into the Running state.

`Stop`

If the migration state is Running, this stops the migration and puts it into the Disabled state.

`Suspend`

If the migration state is Waiting or Running, this puts the migration into the Suspended state.

`Resume`

If the migration state is Suspended, this returns it to Waiting and allows MPS to again begin scheduling migration runs.

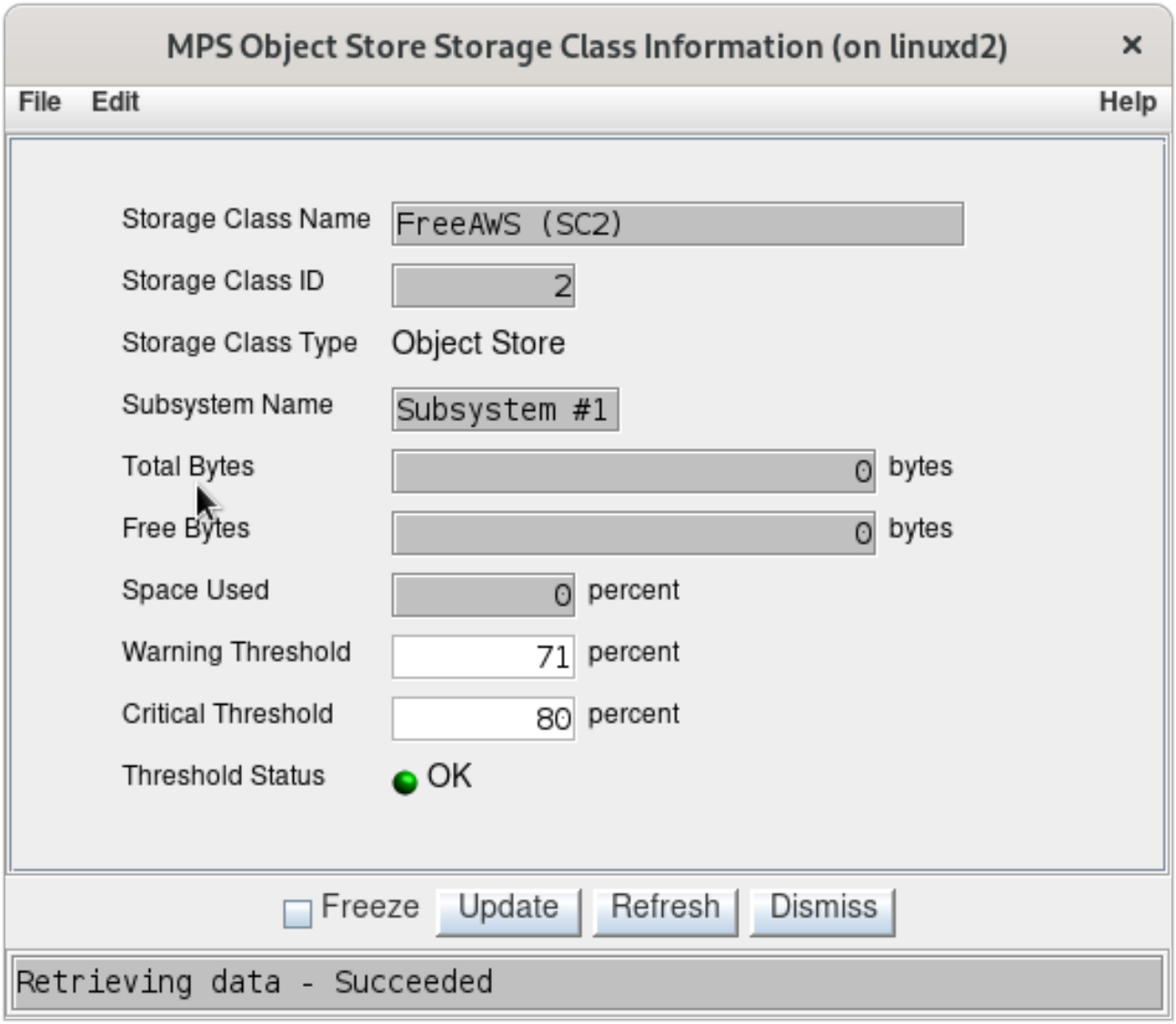
`Reread policy`

Tells the MPS to refresh its migration policy information by rereading the policy.

Pending Operations

When the MPS cannot respond immediately to a control command, a command may be saved as pending. Any such pending operations are displayed here.

19.4.4. MPS Object Store Storage Class Information





What we report here is the total bytes available and the total bytes used. The total bytes used is tracked in the object store volume maps. The total bytes available is based on the value assigned to each object store volume. This value is purely information and provided so that the relative usage can be monitored via the active storage classes information.

Storage Class Name

The name assigned to this storage class.

Storage Class ID

The numeric ID of this storage class.

Storage Class Type

The class of media assigned to this storage class (tape, disk or object store).

Subsystem Name

The name of the storage subsystem for which the storage class information is being displayed.

Total Space

For object store storage classes this reports the total configured capacity of the storage class in bytes.

Free Space

For object store storage classes this reports the unused configured capacity of the storage class in bytes.

Space Used

The percent of configured storage space in the storage class which is used. This is calculated from **Total Space** and **Free Space** and reported as a percentage.

Threshold Status

The current used space threshold status. Possible values are `OK`, `Warning`, and `Critical`.

19.5. Dealing with a space shortage

If free space warning thresholds are exceeded in a storage class, HPSS will warn the administrator with messages in the *Alarms and Events* window. Before the available storage space in a storage class drops to a level where the storage class may no longer be able to satisfy requests, efforts should be made to create additional free space. This can be accomplished by three methods.

The first method is to use the migration and purge functions to free up space. Migration and purge can be run manually; see Manually starting migration and Manually starting purge.

For a long term solution, it may be desirable to tune the migration and purge policies to cause the automatic migration and purge operations to occur more frequently. This method of increasing storage space works for all levels in the Class of Service hierarchy except the lowest level.

The second method is to add additional storage volumes to the storage class using the normal import and create resources procedures. See Adding storage space.

The third method is to increase free tape storage space by repacking sparse volumes and reclaiming empty volumes. The repack(7) program moves information from sparsely filled volumes to other volumes in the same storage class. When the repack(7) is finished, a number of volumes will be cleared of data. To make these volumes ready for reuse, they must be processed by the reclaim(7) program, which refreshes the Core Server metadata that describes the volumes. See Repacking and reclaiming volumes.

19.5.1. Manually starting migration

The Migration/Purge Server runs migration periodically in the time interval specified in the migration policy. However, between these automatic migration runs, an administrator can use the *Active Storage Classes* window to start a migration manually. When a migration run is started manually, the run timer is reset. The next automatic migration will take place after the specified interval after the end of the manual run.

From the *Active Storage Classes* list window, select one or more migratable storage classes and then click on the **Migration Controls** drop-down button and select **Start** from the list.

Note: This request will go straight to the Migration/Purge Server without confirmation.

The Migration/Purge Server stops the migration run when either the Migration Target threshold in the migration policy is reached or there are no more disk files or tape virtual volumes eligible for migration.

The only difference between a manually-started migration run and an automated migration run is that the manually-started run is initiated manually by an administrator. Therefore, a manually-started migration will not migrate files that do not meet the migration criteria.

19.5.2. Manually starting purge

The Migration/Purge Server obtains the disk storage class statistics periodically. The period is set in the Migration/Purge Server's specific configuration record. The Migration/Purge Server then evaluates the used space and may start a purge run. Between these automatic purge runs, an administrator can use the *Active Storage Classes* window to start the purge run manually for one or more storage classes.

From the *Active Storage Classes* window, select one or more purgeable storage classes, and then click on the **Purge Controls** drop-down button and select **Start** from the list.

Note: This request will require confirmation before going to the Migration/Purge Server.

The Migration/Purge Server stops purging when the space used falls to the value specified in the purge policy or there are no more disk files eligible for purging.

The only difference between a manually-started purge run and an automated purge run is that the manually-started run is initiated manually by an administrator. Therefore, a manually-started purge will stop purging when the appropriate threshold is reached just as an automated purge run would.

19.5.3. Repacking and reclaiming volumes

Over time, the active data on a tape virtual volume may become sparse as files are deleted or replaced. The repack(7) utility program provides the administrator with the ability to move data from sparse volumes to other volumes, allowing more efficient use of tape resources. **Repack** is frequently used to migrate files from older tapes to newer tapes when new media replaces older media. The program can be invoked via the command line or from an SSM window. Running repack(7) from the command line affords the greatest flexibility in its use and the opportunity to monitor its progress.

Repack can also move disk data from one volume to another. **Repack** is frequently used to empty a failing disk volume. Begin by setting the volume to RW (or perhaps RO depending on the nature of the failure) using the SSM interface, then move its contents to other volumes using repack(7).

Repack selects tape volumes in one of two ways. The administrator can provide a list of tape volumes to repack, or repack(7) can select volumes based on a number of selection criteria. If repack(7) is provided with a list of tape volumes to process, those volumes must be in RW, RO, EOM or EMPTY condition. Volumes in RWC or DOWN condition cannot be selected. Repacking RW and EMPTY tape volumes is permitted because the administrator has explicitly selected the tape.

Repacking EMPTY tape volumes is also permitted because it is harmless to do so and it provides a way to prove that the volume is empty.

When repack(7) selects tape volumes to process, it uses *number*, *storage class*, *family*, *retired*, *checked out*, and *percentage full* criteria to select a set of tape volumes. The selected volumes will be in RO or EOM condition. Volumes in RWC, EMPTY or DOWN condition are never selected. Volumes that are in RW condition and are also Retired will be treated as if they are in RO condition and may be selected if they qualify under the other selection criteria. If the *--select-retired-tapes* option is used, only retired tapes that meet the rest of the selection criteria will be selected.

When repacking tapes, by default repack(7) will only copy active data to repack output tapes. This means it will skip over inactive data (also called whitespace) on source tapes.

Repack also creates new aggregates on the output tape by bringing together smaller tape storage segments from the input tape and writing new aggregates in the output tape. The *--max-bytes-per -aggregate* command line option controls how this is done. Aggregation can be disabled by specifying `--max-bytes-per-aggregate=0` on the command line. The storage class VV *Estimated Size* is the default value and upper limit for this command line option.

The default behavior of creating new aggregates on the output tape and removing whitespace can be turned off via the *--reduce-tape-positioning* or *--reduce-tape-pos-detect* options. It has been observed that reorganizing the data within a sparse tape (a tape with a significant amount of whitespace) may incur a heavy performance penalty (up to 2-3x) due to extra tape positioning. In certain cases, it may be beneficial to skip this reorganization process in order to achieve faster repack rates. However, it is worth noting that eventually a repack without *--reduce-tape-positioning* should be done to reclaim the whitespace being carried around within the aggregates.

When repacking RAIT volumes, the program accepts the *--select-degraded-rait* command line option. When this option is provided, repack(7) selects tape volumes that contain one or more physical volumes that have had mount, read or write errors. Specific details regarding such errors are recorded in metadata and are accessible via the lspvhist(7) utility program. This program can help to shed further light on whether an apparently degraded RAIT volume should be repacked.

Tape repack(7) can segregate repack output tapes from other tapes. When this feature is enabled, repack(7) will copy files only to tapes that contain repack output. New files migrated to tape will be written on other tapes. This feature is controlled by the HPSS_CORE_REPACK_OUTPUT environment variable. If this variable is present in the system's environment, and if its value is "on", tape repack output tapes will be segregated from other tapes. This feature is "on" by default. Set HPSS_CORE_REPACK_OUTPUT to "off" to disable it system-wide. As an alternative to disabling the feature system-wide, this feature can be turned off for a specific repack(7) instance. This is controlled by repack(7)'s `--no-dest-tape-segregation` command line option. See the repack man page for more details.

Disk volumes to be repacked must be in RW, RO or EMPTY condition. EMPTY volumes may be repacked because it is harmless to do so, the administrator has explicitly selected the disks, and it provides a way to prove that the volume is empty.

Repacking disks cannot be done via the SSM window; the program must be executed from a command line following the instructions given in the repack(7) man page.

When the repack(7) process completes successfully, the Number of Active Segments in the volume's display will be zero and the VV Condition will be EMPTY.

Note that tape volumes in EOM condition go to EMPTY condition when repack(7) moves all of the data on the volumes to other volumes. Volumes in RW and RO condition do not go to EMPTY condition. Manually setting the condition of an empty tape volume to EOM causes it to go to EOM condition, then immediately to EMPTY condition.

After a tape volume is successfully repacked, it remains in EMPTY state and is not available for storing data. To make the volume available, it must be reclaimed. Disk volumes can be placed back in service by changing the VV Condition to RWC.

Note that both disk and tape repack operations can put significant demands on system resources, particularly when repacking striped volumes. Since repack(7) performs a tape-to-tape copy operation, it will consume twice as many tape drives as the stripe width of the volumes being repacked.

The reclaim(7) program deletes empty tape virtual volumes and re-creates them with the same configuration as the original volume. The PV Name(s), Stripe Width, PV and VV Block Sizes, PV and VV Estimated Size and other parameters of the old VV are used to create a new VV with a new Virtual Volume ID (VVID). The metadata describing the old VV is deleted. HPSS requires that tapes be reclaimed, rather than reused, so that it can be certain that no references to the old VV exist or could be resolved.

Reclaimed tape volumes are entered into HPSS metadata in RWC condition and are immediately available for use.

19.5.4. Verifying data integrity

An administrator may want to verify data which has been sitting at rest on a tape volume periodically or in response to some event. This can be accomplished by using the **lbp_verify** tool.

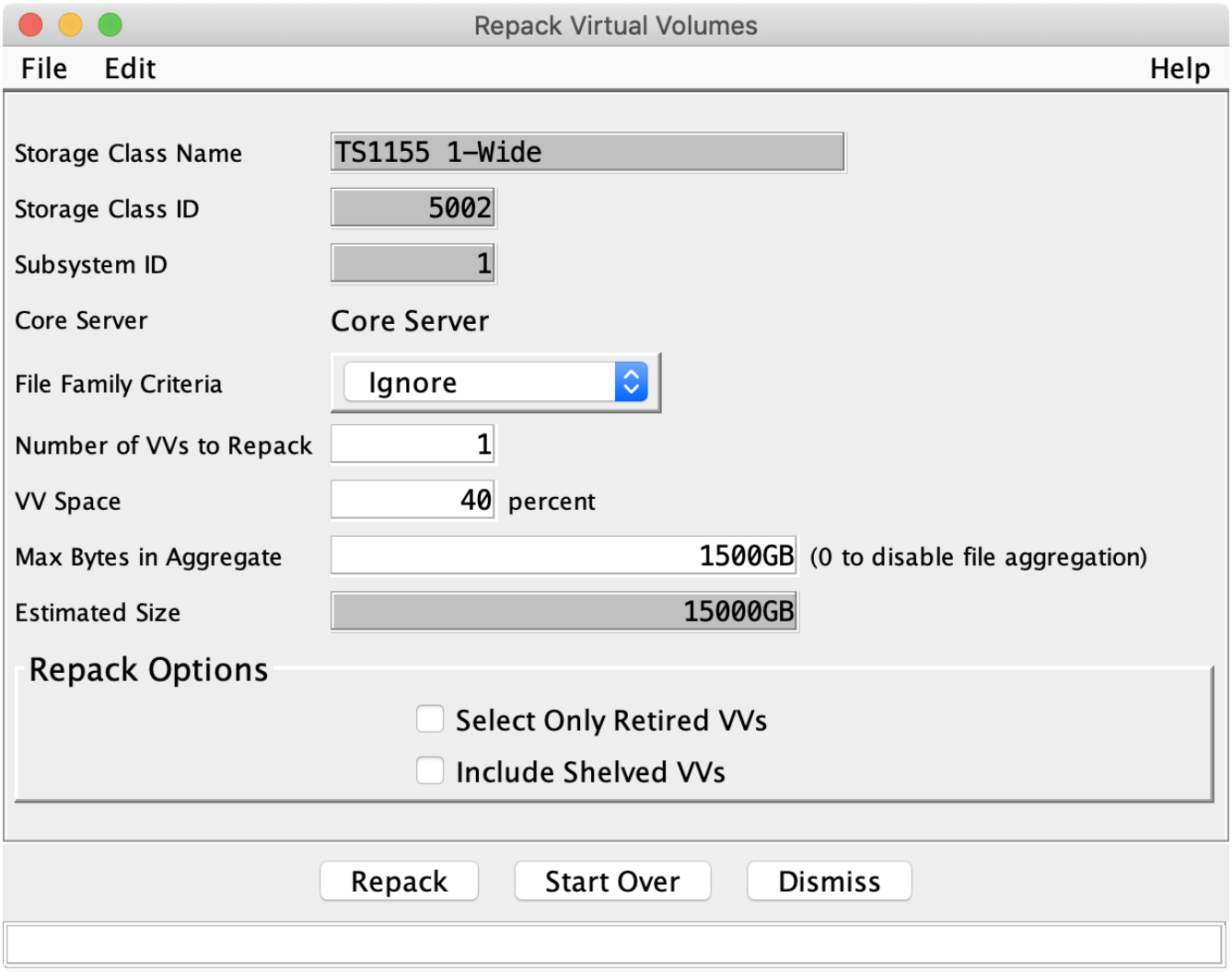
lbp_verify causes a tape volume to be mounted, and segments read and verified against a CRC stored on the medium. The tool provides a number of options for specifying inputs and drive resources to use. See the `lbp_verify.7` man page for more details.

While the tool is verifying the volume, no other I/O will be processed against the volume. When the verification is complete, an entry will be logged in the PV history table and if all PVs in the VV were successfully verified, the volume's last verify time will be updated. If an error is found, information about the location of the error will come back from the tool and be logged in the PV history table.

Verification uses a checkpoint system to keep track of progress against a tape. If for some reason verification cannot be completed, the next time the volume is requested to be verified, the process can be started again from the checkpoint.

If enough device errors occur during a verify operation, HPSS will prematurely halt the verify. The HPSS_MOVER_VERIFY_ERROR_THRESHOLD environment variable controls how aggressively HPSS will halt verify operations in response to device errors. See the `lbp_verify.7` man page for more details.

19.5.4.1. Repack Virtual Volumes window



This window provides an interface for repacking tape virtual volumes.

Field descriptions

Storage Class Name

The name of the storage class that will be repacked.

Storage Class ID

The ID of the storage class that will be repacked.

Subsystem ID

The ID of the subsystem which contains the volumes to be repacked.

Core Server

The name of the Core Server that manages the volumes to be repacked.

File Family Criteria

Indication of whether to use file family criteria. Click on the button to drop-down a list of available options. Select `Ignore` to indicate not to use the file family criteria when selecting volumes for repack. Select the name of a file family to repack if you want to use file family

criteria. When a file family is selected, volumes assigned to that family will become candidates for repack. Volumes not in the selected family will not be candidates. Storage segments on repacked volumes will be transferred to other volumes assigned to the same family. If no writable tape volumes are available in the family, a new blank volume will be chosen and assigned to the family. If `Not in a family` is selected, volumes not assigned to any file family will become candidates for repack. Volumes that have been assigned to a file family will not be candidates.

Number of VVs to Repack

The maximum number of tape volumes to repack. If repack(7) does not find this many candidate volumes, those that have been selected will be repacked and no error will result.

VV Space

This is a repack selection criterion expressed as a percentage. If the amount of occupied space on a candidate tape is less than this value, the tape will be considered for repacking. Tape volumes with larger amounts of occupied space will not be considered.

The occupied space percentage is calculated by summing the lengths of the files on the candidate tape and comparing that sum to the VV Estimated Size field in the storage map. If this ratio is less than the **VV Space** percentage, the volume is selected. Note that vacated files in file aggregates on tapes will not be counted as occupied space. A tape containing a relatively small number of large file aggregates from which a large percentage of the files have been deleted, may be selected by repack(7).

If **VV Space** is 100, the comparison is not performed and selection of tape volumes to repack is made using the remaining criteria.

Max Bytes in Aggregate

This sets a limit on the number of bytes that will be accumulated into a tape aggregate written by repack(7). This value is limited to the configured **Estimated Size** of VV's in the selected storage class. Set this value to zero to disable the creation of new aggregates in output tapes. This value has no effect when repacking disks.

Estimated Size

The estimated size of the storage class to be repacked. This value comes from the storage class configuration and is for informational purposes only to help in setting a valid **Max Bytes in Aggregate** value.

Repack options

Select Only Retired VVs

If selected, repack(7) selects only retired volumes in the indicated storage class. Retired volumes may be in RO, EMPTY or EOM condition. DOWN volumes are never selected. If this option is not selected, both retired and non-retired tape volumes in RO, EMPTY or EOM condition may be selected.

Include Shelved VVs

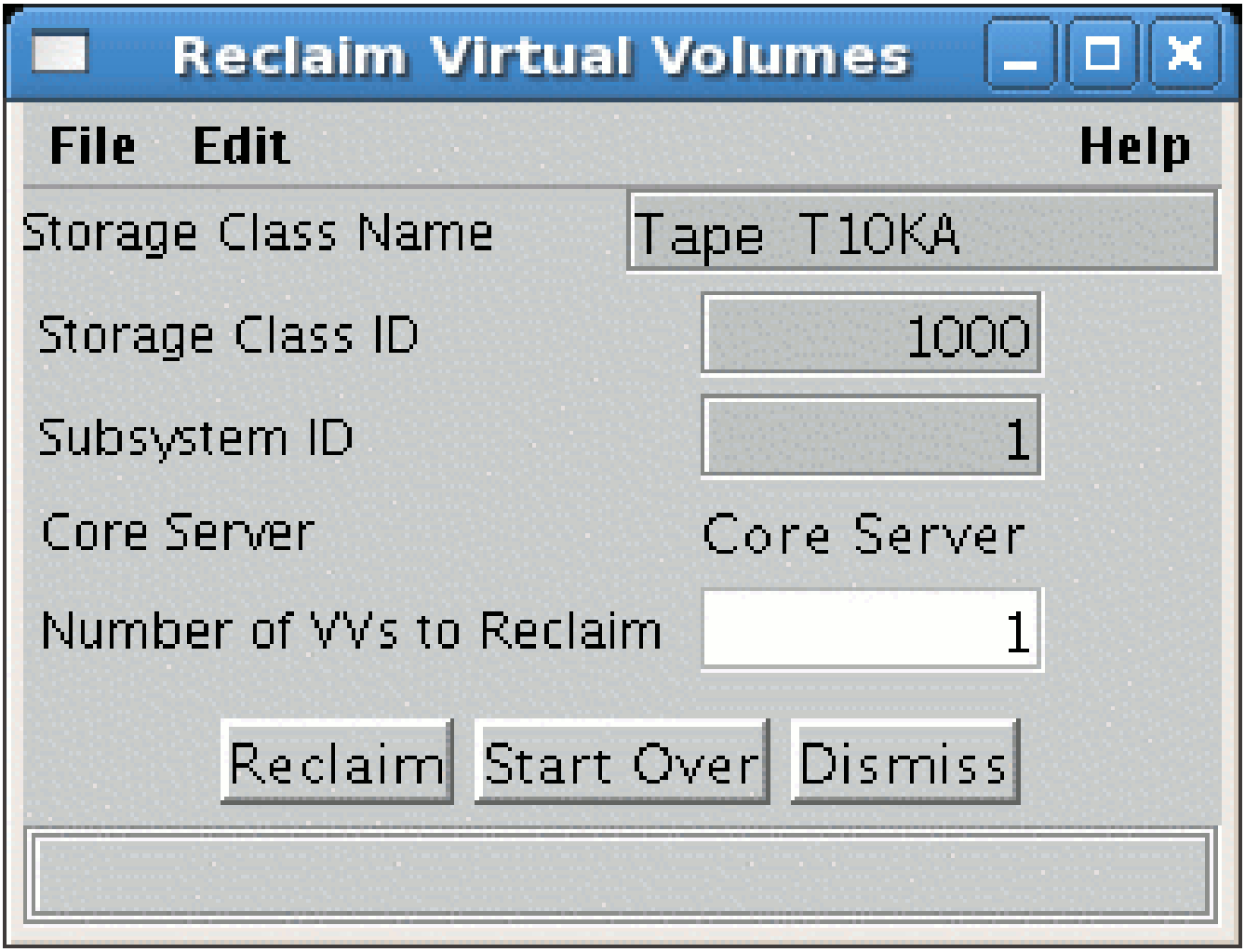
Tape volumes which have been removed from HPSS using the shelf tape utility are eligible for repack.

Buttons

Repack

Click this button to start the repack(7) program on the indicated storage class. Tape volumes that meet the selection criteria will be repacked. Status messages are displayed on the status bar at the start and end of the repack.

19.5.4.2. Reclaim Virtual Volumes window



This window is used to run the reclaim(7) utility program which removes and recreates tape virtual volumes.

Field descriptions

Storage Class Name

The name of the storage class selected for reclaiming.

Storage Class ID

The ID of the storage class selected for reclaiming.

Subsystem ID

The ID of the subsystem in which the volumes to be reclaimed reside.

Core Server

The name of the Core Server that will perform the reclaim operations.

Number of VVs to Reclaim

The number of tape virtual volumes in the storage class that will be reclaimed. If the number of candidate volumes is less than this value, all eligible volumes will be reclaimed. Retired volumes are never reclaimed.

Buttons

Reclaim

Click this button to start the reclaim(7) utility program on the indicated storage class. Tape volumes that are described as EMPTY, and are not retired, will be reclaimed. Status messages are displayed on the status bar at the bottom of the window at the start and end of the reclaim.

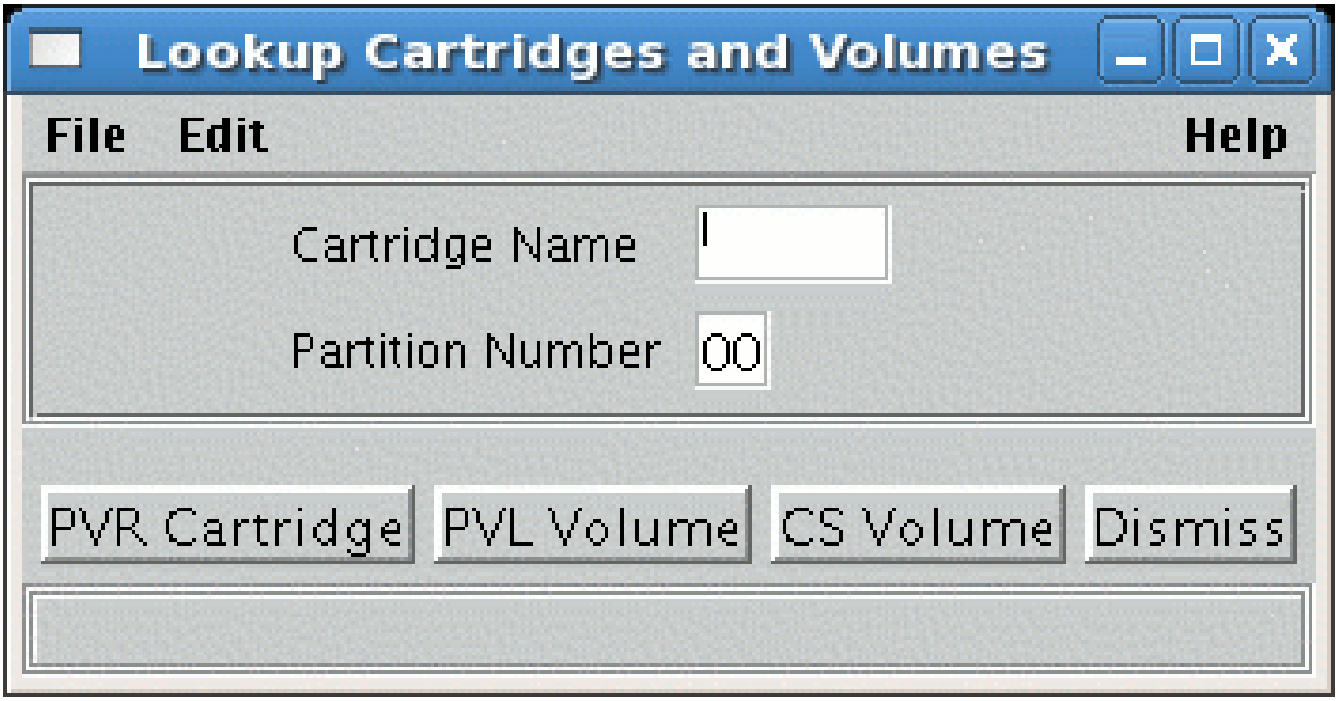
SSM invokes the reclaim(7) utility program and passes it the storage class and number of tape volumes to be reclaimed, as entered on the *Reclaim Virtual Volumes* window. The utility program selects empty tape volumes from the storage class until it selects the required number or the supply of empty volumes is exhausted. The volumes are reclaimed and immediately become available for use.

The reclaim(7) utility can also be invoked as a command line utility. Used this way, the user has more flexibility in choosing options and can monitor the program's progress.

19.6. Volume management

This section describes how to monitor and manage PVL volumes, PVR cartridges, and Core Server disk and tape volumes.

19.6.1. Lookup Cartridges and Volumes window



This window allows you to access information windows for PVR cartridges, PVL volumes, and Core Server volumes. After clicking a button to perform a lookup operation, this window will remain open to allow additional lookups.

Field descriptions

Cartridge Name

The six-character cartridge label. A string exactly six characters long must be entered in this field.

Partition Number

This field is no longer used and should always be "00".

Buttons

PVR Cartridge (tape only)

Once you have filled in the cartridge name, clicking on this button will open the *PVR Cartridge Information* window for the specified cartridge. This metadata is created when the cartridge is successfully imported.

PVL Volume

Once you have filled in both fields, clicking on this button will open the *PVL Volume Information* window for the specified volume. This metadata is created when the volume is successfully imported.

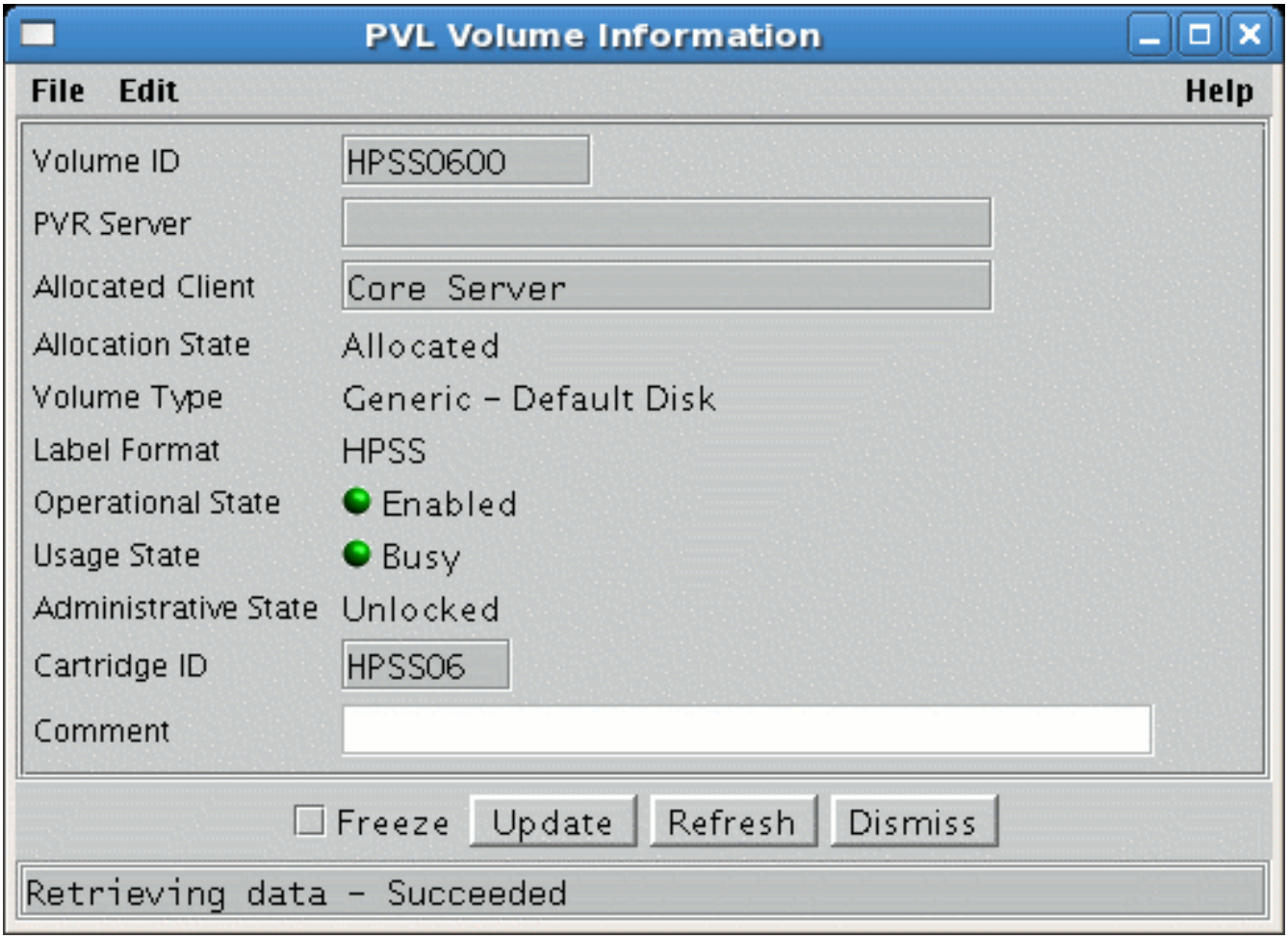
CS Volume

Once you have filled in both fields, clicking on this button will open the *Core Server Disk Volume* or *Core Server Tape Volume* window for the specified volume. This metadata is created when the disk or tape storage resources are successfully created.

19.6.2. PVL Volume Information window

The *PVL Volume Information* window allows the SSM user to view the data for imported volumes.

Before using the window, the user should know the six-character label of the PVL volume. The dumppv_pvl(7) utility can be used to list all the volumes being managed by the PVL and help determine the label.



This window allows you to view the information associated with a PVL volume.

Field descriptions

Volume ID

The eight-character volume label for the volume being viewed.

PVR Server

The descriptive name of the PVR that owns the cartridge corresponding to the volume being viewed. This field is applicable only to tape volumes and is always blank for disk volumes.

Allocated Client

The descriptive name of the Core Server which manages the volume. This is the name of the Core Server specified when storage resources were allocated on the volume using the Create Disk Resources or Create Tape Resources operation. If no storage resources have been created on the volume, this field will be blank.

Allocation State

The usability of the volume. This includes whether the volume import and label operation completed successfully, whether storage resources have been created on the volume, and whether the volume is currently stored in the robot or removed to an external shelf or vault.

Possible values are:

- `Allocated` indicates storage resources have been created on the volume, it is allocated to a Core Server, and it is available to the system for I/O.
- `Unallocated` indicates the volume has been successfully imported into the HPSS system and labeled. However, no storage resources have been created on it and it has not been allocated to a Core Server. It is therefore not available to the system for I/O.
- `Allocated - On Shelf` indicates storage resources have been created on the volume, it is assigned to a Core Server, and it is available to the system. However, it has been removed from the tape robot (to a vault, for example), so there will be a delay for any mount operation while the volume is returned to the robot.
- `Unlabeled` indicates an error occurred during the mount/label operation during import. A volume in this state is not usable by HPSS. It should be exported and re-imported.
- `Allocated - No Shelf Info` indicates the volume has been allocated to a Core Server and is available to the system for I/O, but no shelf information is available because the PVL can not communicate with the PVR.

Volume Type

The HPSS media type corresponding to this volume.

Label Format

The type of label on the volume. Possible values are:

- `HPSS` indicates the volume has an ANSI label (it starts with an 80-byte block beginning with the characters "VOL1"). The owner field of the label contains "HPSS".
- `Foreign` indicates the volume has an ANSI label, but the owner field is something other than "HPSS".
- `None` indicates the volume has no label. A volume in this state is not usable by HPSS. It should be exported and re-imported.
- `Data` indicates the volume is an uninitialized tape.
- `NonANSI` indicates the label is an 80-byte non-ANSI record.
- `NSL` indicates the volume has a NSL label format (80-byte record without EOF).
- `Unknown` indicates the label of the volume failed. Status of label unknown.

Operational State

This will always be `Enabled`.

Usage State

`Busy` if the volume is in use; `Idle` otherwise.

Administrative State

This will always be `Unlocked`.

Cartridge ID

The six-character cartridge label for the volume.

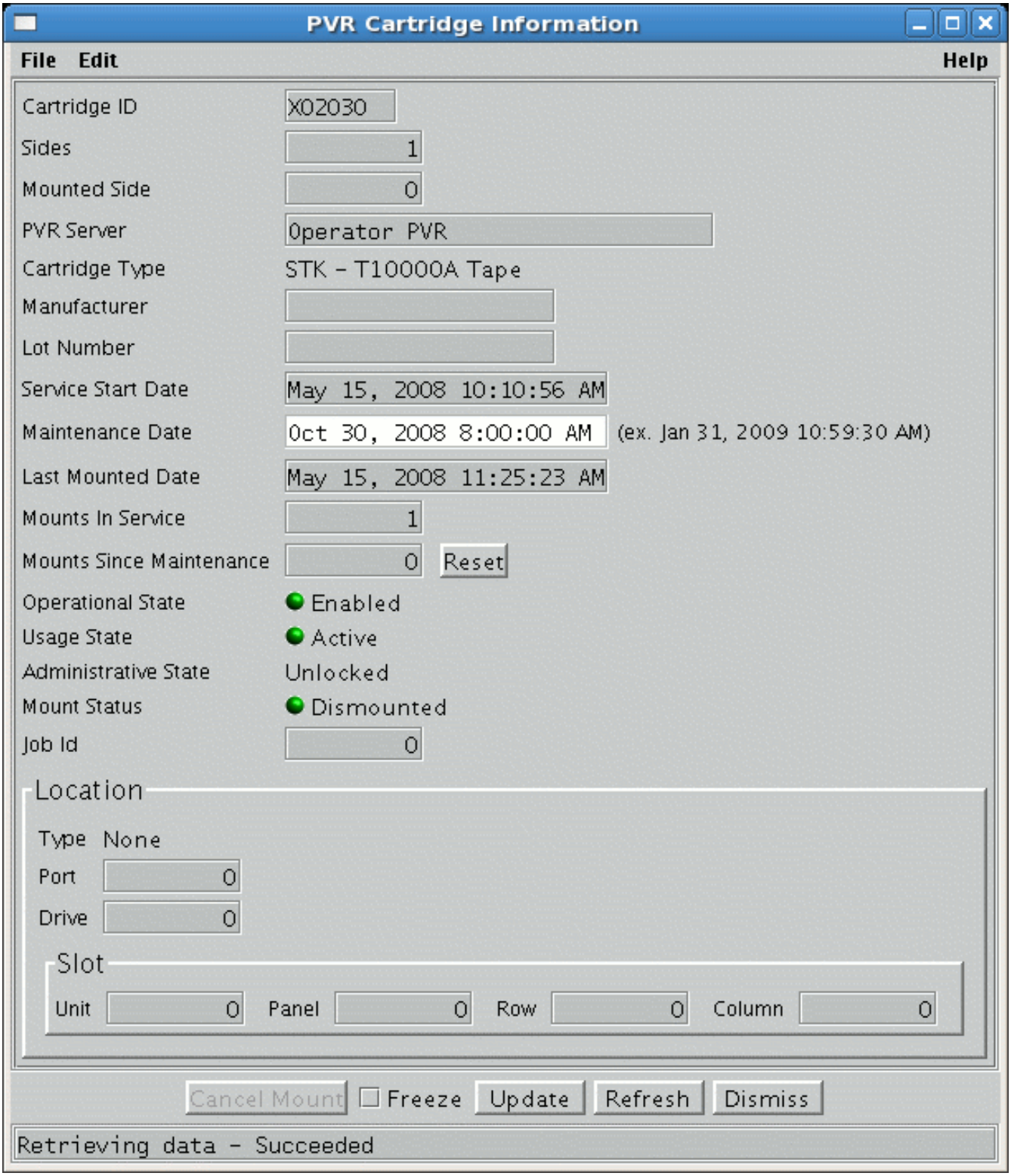
Comment

This field provides a 128-character buffer in the PVL volume metadata which gives the administrator the opportunity to associate miscellaneous text with a volume. For example, a site may want to place a comment in this field that the volume (cartridge or disk) is out of service.

19.6.3. PVR Cartridge Information window

The *PVR Cartridge Information* window allows the SSM user to view the data about a cartridge managed by one of the HPSS PVRs.

Before using the window, the user should know the six-character label of the cartridge. The dumppv_pvr(7) utility can be used to list all the cartridges being managed by the PVR and help determine the label.



This window allows you to view and update the information associated with an HPSS tape cartridge.

Note that the **Location Type** fields are represented differently for certain types of robots, for which **Port**, **Drive** and **Slot** (**Unit**, **Panel**, **Row**, and **Column**) may each be displayed as "0". If it is necessary to locate a cartridge in one of these robots, the robot's operator interface must be used.

This window contains three bookkeeping fields: **Maintenance Date**, **Mounts Since Maintenance**, and **Mount Status**. If the some maintenance is performed on the cartridge, such as re-tensioning it, the **Mounts Since Maintenance** field can be reset. The **Maintenance Date** field can also be reset. Note that resetting one field does not affect the other. The third operation involves resetting the

Mount Status from `Mount Pending` to `Dismounted` via the **Cancel Mount** button. This operation is only required for handling a specific error condition.

Any changes made to fields on this window are sent directly to the PVR and are effective immediately.

Field descriptions

Cartridge ID

The six-character cartridge label.

Sides

The number of partitions (sides) on the cartridge. Currently always set to "1".

Mounted Side

The partition currently mounted. Set to zero if the cartridge is not currently mounted; note that this can also mean that side 0 of the cartridge is mounted.

PVR Server

The descriptive name of the PVR which manages the cartridge.

Cartridge Type

The HPSS media type corresponding to this cartridge. This controls which the type of drive in which the cartridge can be mounted.

Manufacturer

The Manufacturer string specified when the cartridge was imported.

Lot Number

The Lot Number string specified when the cartridge was imported.

Service Start Date

The date and time when the cartridge was imported.

Maintenance Date

If the maintenance is performed on the cartridge, for example by retensioning it, this field can be set to record the date of the maintenance. The format for entering the date is given to the right of the field. This format can be changed with the **-D** option to the hpssgui(7) startup script.

Last Mounted Date

The date and time when the cartridge was last mounted.

Mounts In Service

The number of times the cartridge has been mounted by HPSS since it was imported.

Mounts Since Maintenance

The number of times the cartridge has been mounted since its last maintenance. Click the **Reset** button to the right of this field to reset the value to zero. This field is not automatically reset

when Maintenance Date is changed.

Operational State

This will always be `Enabled`.

Usage State

This should always be `Active`.

Administrative State

This will always be `Unlocked`.

Mount Status

The mount status of the cartridge. Possible values are:

- `Mount Pending`
- `Mounted`
- `Dismount Pending`
- `Dismounted`
- `Eject Pending`
- `Checked In`
- `Check-In Pending`
- `On Shelf`
- `Check-Out Pending`
- `Move Pending`
- `Move Shelf Pending`

Job Id

The job identifier associated with the cartridge. A value of zero indicates that there is not a job associated with the cartridge (that is, when the cartridge is not mounted).

Location Type

The type of cartridge location information being displayed. Possible values are:

- `None` indicates no location information is available.
- `Port` indicates the location is a port number. [This option is currently not used.]
- `Drive` indicates the location is a drive ID number.
- `Slot` indicates the location is a slot specification.

The following fields are filled in (nonzero) based on the **Location Type** and whether or not the PVR has the information:

Port

The port number where the cartridge is located. This option is currently not used and thus will always be zero.

Drive ID

The drive ID number where the cartridge is located. This field is valid when the **Location Type** is Drive. It is used/valid when the cartridge is Mounted; it is not used/valid (and thus will have the value of zero) when the cartridge is not Mounted.

Slot

The slot address (**Unit**, **Panel**, **Row** and **Column**) for the cartridge. These fields are valid only if **Location Type** is Slot and the PVR supports this (STK). The fields will be zero when it's not valid.

Buttons

Cancel Mount

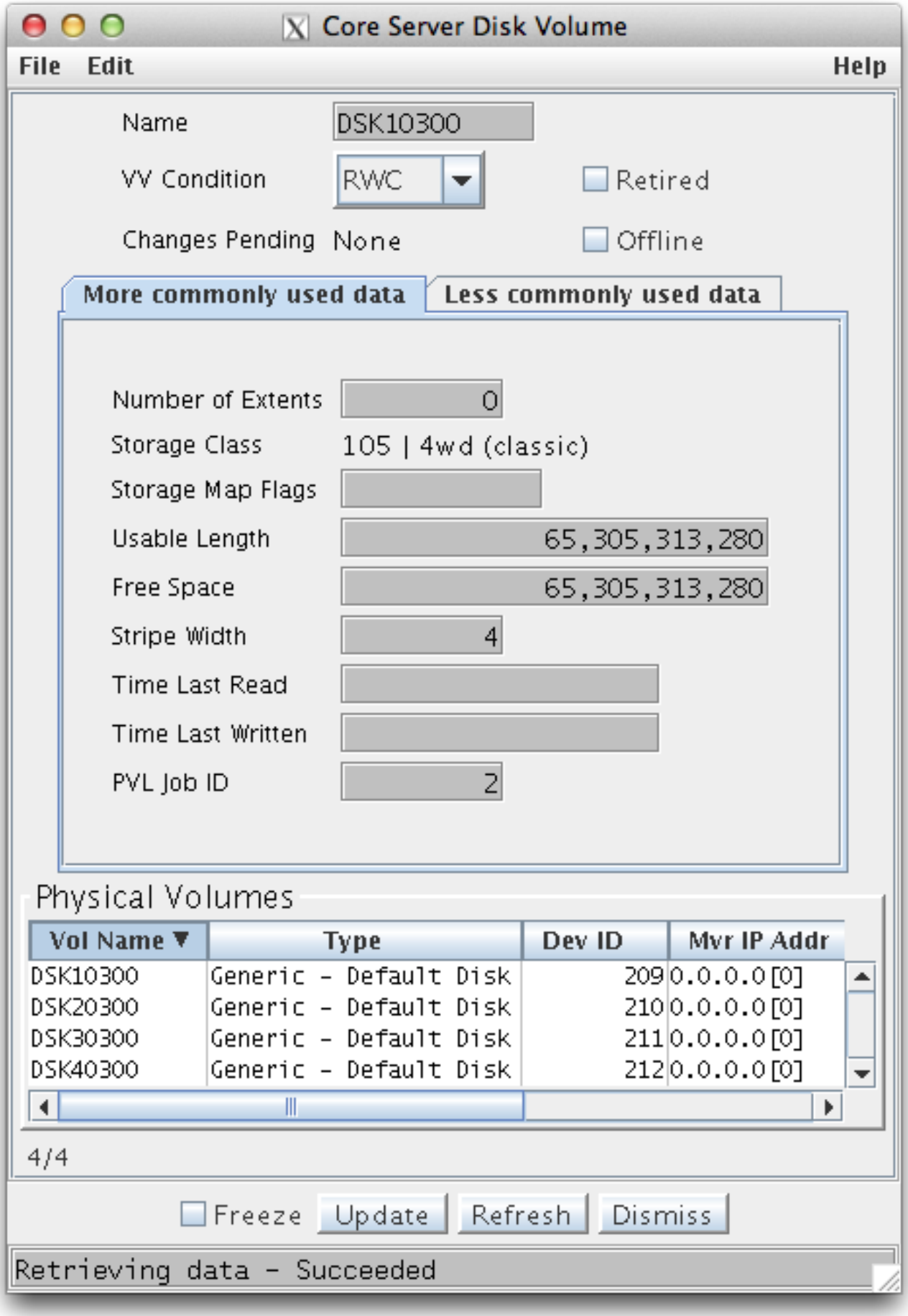
When **Mount Status** is displaying `Mount Pending`, the **Cancel Mount** button to the right of the field will be sensitized. If you click the button, a large warning will be displayed, and you will have to confirm your request by clicking **Cancel Mount** again. This will send a request to the PVR to cancel the mount request for the cartridge. As noted in the on-screen warning, this should only be done as a last-ditch attempt to cancel a mount request which cannot be canceled in any other way.



Note that this will not dismount a mounted tape.

19.6.4. Core Server Volume and Segment windows

19.6.4.1. Core Server Disk Volume information window



This window displays information about a disk volume as represented by the Core Server.

Name

The ASCII name of the first physical volume that is a part of the disk virtual volume. The entire virtual volume can be referred to by this name.

VV Condition

This is the administrative control for the disk virtual volume. It will have one of five values: RWC, RW, RO, EMPTY or DOWN.

- In RWC condition, the volume can be read and written. This is the normal operational state.
- In RW condition, the volume can be read and written, but new disk storage segments may not be created on the volume.
- In RO condition, the volume can be read but not written. New storage segments cannot be created on the volume.
- In EMPTY condition, the volume cannot be read or written. All storage segments have been removed from the volume, and new segments cannot be created. Since this condition represents a physical state of the volume, the operator may not change the volume to this condition. EMPTY is displayed by the server when the condition is RO and there are no storage segments in the volume.
- In DOWN condition, the volume cannot be read, written or mounted. This condition can be used to make a disk unavailable to the system.

Change the **VV Condition** of a disk virtual volume by selecting the desired condition from the drop-down list and then clicking the **Update** button.

Changes Pending

If there are any VV Condition changes for this volume pending in the Core server, **Changes Pending** will be `Yes` with a red bullet. Otherwise, **Changes Pending** will be `None`.

Retired

When checked, the volume is retired. New storage segments will not be created on the volume.

Offline

When checked, the volume is considered offline and not usable for reading nor writing. This often occurs as a result of failed communication with the Mover that controls the volume. HPSS will attempt to automatically return offline disks to service. For disks that HPSS can automatically set back online, you will see corresponding entries in the Alarms and Events window. Because this occurs in the background, failures to automatically set disk volumes back online are silent.

If a disk is functioning sporadically, the disk may transition back and forth between offline and online. If you want to take the disk out of service for repairs, set its VV Condition to DOWN.

A reminder about offline disks will appear in the hpssgui(7) *Alarms and Events* window on a regular basis. By default, the reminder message appears once a day in the mid-morning. You can alter the frequency with which the message appears by setting the

HPSS_OFFLINE_DISK_REMINDER_INTERVAL_HOURS environment variable. Note that the value you specify should be a number of hours, not minutes nor seconds.

Offline versus DOWN

Disks that are either offline or have a VV Condition of DOWN are unusable for read and write activity. They are essentially *out of service*. Though both states, offline and DOWN, achieve the same end result, they reflect different situations.



The offline state reflects a physical reality (for example, the disk's Mover may be down or not reachable over the network). It is automatically managed by HPSS and is not meant to be used to administratively control access to disk volumes. In general, if you want a disk to be unusable, *set its VV Condition to DOWN rather than setting the **Offline** state*.

DOWN is an administrative state. It's useful for removing a disk from service regardless of whether it has an apparent malfunction.

More commonly used data tab

Number of Extents

The number of disk storage segment extents allocated in this virtual volume.

Storage Class

The storage class to which the disk virtual volume is assigned.

Storage Map Flags

This field is usually blank for volumes that are actively being used to store incoming data. Volumes in other states (such as, but not limited to, full tapes, tapes with no data yet written to them, tapes written to by repack, offline disks) may contain a status in the field. This field is informational only.

Usable Length

The amount of disk space that can be used for storing storage segments. This is the Actual Length of the disk volume minus certain amounts set aside for system use.

Free Space

The number of bytes on the disk virtual volume that are not assigned to storage segments. **Free Space** is initially set to the value of **Usable Length**, then decremented and incremented by the size of disk storage segments as they are created and deleted. Since disk storage segments are allocated in whole clusters, they are usually larger than the amount of storage needed to store the segment of the file. This creates "slack space" which is space allocated to files, but not filled with file data. The amount of disk space in use for any volume, which is the **Usable Length** minus the **Free Space**, will always be somewhat greater than the sum of the lengths of the file segments stored on the volume. Since unallocated disk space may become fragmented, it may not be possible to create new disk storage segments on a volume even if the amount of free space appears to be large. The size of the largest free extent on the volume is logged by the Core Server when this happens.

Stripe Width

The number of physical volumes in the stripe group that makes up the virtual volume. All volumes in HPSS are considered to be striped, even if this value is one.

Time Last Read

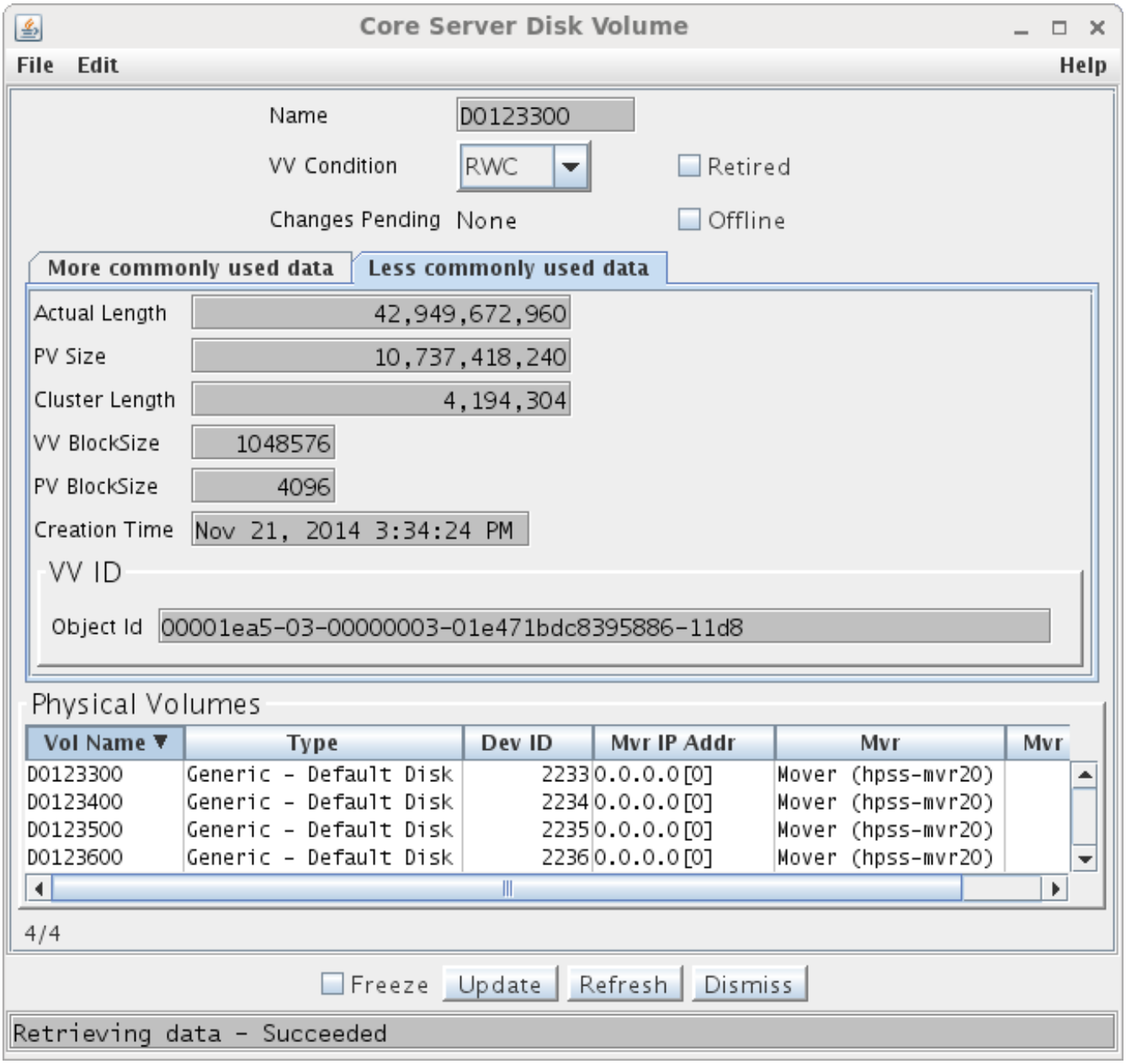
The date and time the volume was last read. This value is not stored in metadata, but is initialized to zero when the Core Server starts and then updated each time the disk is read. This field is not affected by transaction aborts.

Time Last Written

The date and time the volume was last written. This value is not stored in metadata, but is initialized to zero when the Core Server starts and then updated each time the disk is written. This field is not affected by transaction aborts.

PVL Job ID

The PVL job ID in which this volume is mounted. This field will be zero if the volume is not mounted.



Less commonly used data tab

Actual Length

The length of disk virtual volume in bytes. This length includes all of the space set aside for system use. See **Usable Length**.

PV Size

The length in bytes of a physical volume in this disk virtual volume. All physical volumes in a disk virtual volume must be the same length.

Cluster Length

The size of the allocation unit, in bytes. When disk storage segments are created on the volume, they are created at a length that will be a multiple of this value.

VV Block Size

The virtual volume block size. This is the number of bytes written from a data stream to an element of the striped volume before the stream switches to the next element of the stripe.

PV Block Size

The size, in bytes, of the media data block.

Creation Time

The date and time the disk volume was created in the Core Server.

VV ID

The detailed description of the disk virtual volume ID associated with this disk virtual volume.

Physical Volumes

This is a table of physical volume attributes for the physical volumes that make up this disk virtual volume.

Vol Name

The ASCII name of the physical volume.

Type

The media type.

Dev ID

The ID of the device the physical volume is mounted on.

Mvr IP Addr

The IP address of the Mover that operates this physical volume.

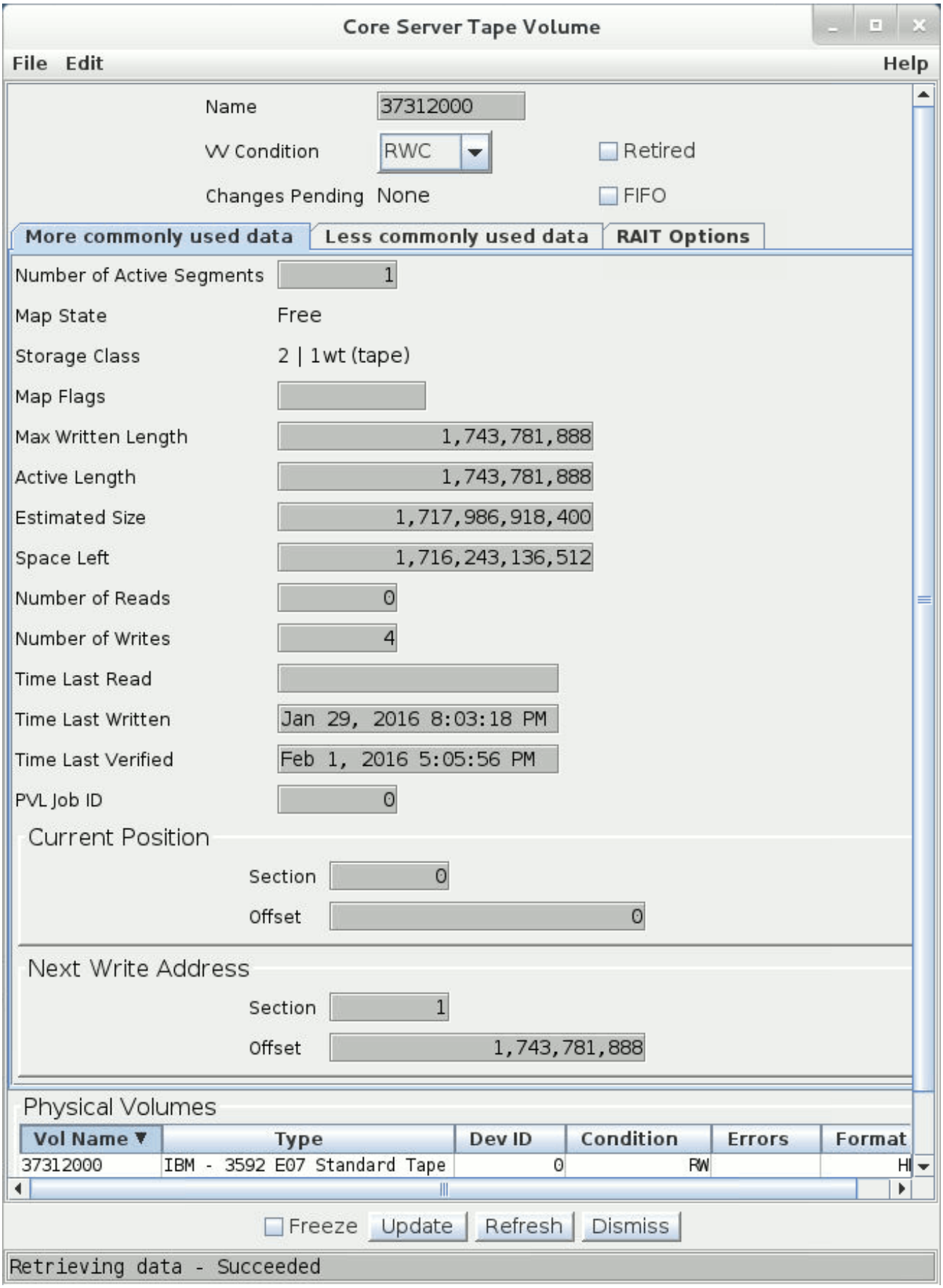
Mvr

The descriptive name of the Mover that operates this physical volume.

Mvr Host

The name of the host on which the Mover runs.

19.6.4.2. Core Server Tape Volume information window



This window displays information about a tape volume as represented by the Core Server.

Field descriptions

Name

The ASCII name of the first physical volume that is a part of the tape virtual volume. The entire virtual volume can be referred to by this name.

VV Condition

This is the administrative control for the tape virtual volume. It will have one of five values: RWC, RO, EOM, EMPTY or DOWN.

- In RWC condition, the volume can be read and written. This is the normal operational state.
- In RO condition, the volume can be read but not written. New storage segments cannot be created on the volume.
- In EOM condition, the volume can be read but not written. One or more of the tapes has been written to its end and the tape virtual volume is now full. New storage segments cannot be created on the volume. The volume condition can be changed to DOWN, but not to any other condition.



Note that setting a volume to EOM is irreversible. Don't use this setting unless you're sure it's what you want.

- In EMPTY condition, the volume cannot be read or written. The volume has reached EOM, and all storage segments have been removed from the volume. New storage segments cannot be created. Volumes cannot be changed manually to EMPTY condition. EMPTY is reported by the Core Server when the volume has reached EOM and no storage segments remain on the volume.
- In DOWN condition, the volume cannot be read, written or mounted. This condition can be used to make a tape unavailable to the system.

Change the condition of a tape virtual volume by selecting the desired condition from the drop-down list and then clicking the **Update** button.

Changes Pending

If there are any VV Condition changes for this volume pending in the Core server, **Changes Pending** will be `Yes` with a red bullet. Otherwise, **Changes Pending** will be `None`.

Retired

When checked, the volume is retired. New storage segments will not be created on the volume. Retired volumes may not be reclaimed.

FIFO

When checked, the Core Server will schedule read requests on the volume in FIFO order, regardless of the Core Server's tape queue method settings. The VV's queue will not become FIFO-ordered until the next time the queue is ordered — generally due to a new request coming in.

More commonly used data tab

Number of Active Segments

The number of storage segments stored in this virtual volume.

Map State

This is the primary operational state flag of the tape virtual volume. It can take on one of the following values:

- `Free` indicates the volume is available to be written.
- `Allocated` indicates the volume is assigned to a tape write operation and is being written.
- `EOM` indicates the volume has reached End Of Media and can no longer be written.
- `Empty` indicates the volume has reached End Of Media and all storage segments have been deleted.
- `Deny` indicates the volume is not eligible to have a new tape storage segment created on it. Tape read operations are permitted, and the last storage segment on the tape may be written under certain circumstances.

The **Map State** field is displayed to provide an additional level of detailed information about the tape volume. It cannot be changed manually. Use the **VV Condition** to change the operational condition of the volume.

A tape volume starts in `Free` state, which indicates that it can be selected to receive a storage segment and its data. When data is being written to a volume, it will be shown in `Allocated` state. When the End Of Media marker is reached during a tape write operation, the volume will enter `EOM` state. If the number of segments on the volume drops to zero after reaching EOM, the state will change to `Empty`. The volume may be placed in `Deny` state by the server depending on the setting of the VV Condition.

Storage Class

The storage class to which the tape virtual volume is assigned.

Map Flags

Information about the state of the volume. Contains one of the following values:

- `Never Written` indicates data has not been written to the volume.
- `Retired` indicates new data will not be written to this volume.
- `Repack Output` indicates all files on this volume were written there by the repack process.

Max Written Length

The total number of bytes that have been written to the volume. Each time the volume is written, this number is incremented by the number of bytes written. This number is not decremented when storage segments are deleted from the volume. This value does not include parity bytes in RAIT volumes.

Active Length

The sum of the number of bytes in active storage segments. As segments are added to the

volume, this number increases. As segments are deleted, this number decreases. This value does not include parity bytes in RAIT volumes.

Estimated Size

The estimated size of the volume in bytes. It is only an estimate of the number of bytes that can be written on a tape volume. Depending on the compression characteristics of the data, the actual number of bytes may be smaller or greater. This value does not include parity bytes in RAIT volumes.

Space Left

The estimated number of bytes that may still be written to the tape virtual volume. It is initially set to **Estimated Size** and is decremented as storage segments are written. It is not incremented when storage segments are deleted. This value does not include parity bytes in RAIT volumes.

Because of the variable compressibility of data written to tapes, this value may behave strangely. For instance, the end of the media (EOM) may be reached earlier than estimated, causing this value to become zero quite suddenly. Conversely, EOM may be reached later than expected. In this case, **Space Left** will be decremented until it reaches a value of "1", then remain at "1" until EOM is reached.

Number of Reads

The number of times the tape volume has been read.

Number of Writes

The number of times the tape volume has been written.

Time Last Read

The date and time the volume was last read. If this field is blank, the tape volume has not been read.

Time Last Written

The date and time the volume was last written. If this field is blank, the tape volume has not been written.

Time Last Verified

The date and time the volume was last verified (via the lbp_verify(7) tool). If this field is blank, the tape volume has never been verified.

PVL Job ID

The PVL job ID in which this volume is mounted. This field will be zero if the volume is not mounted.

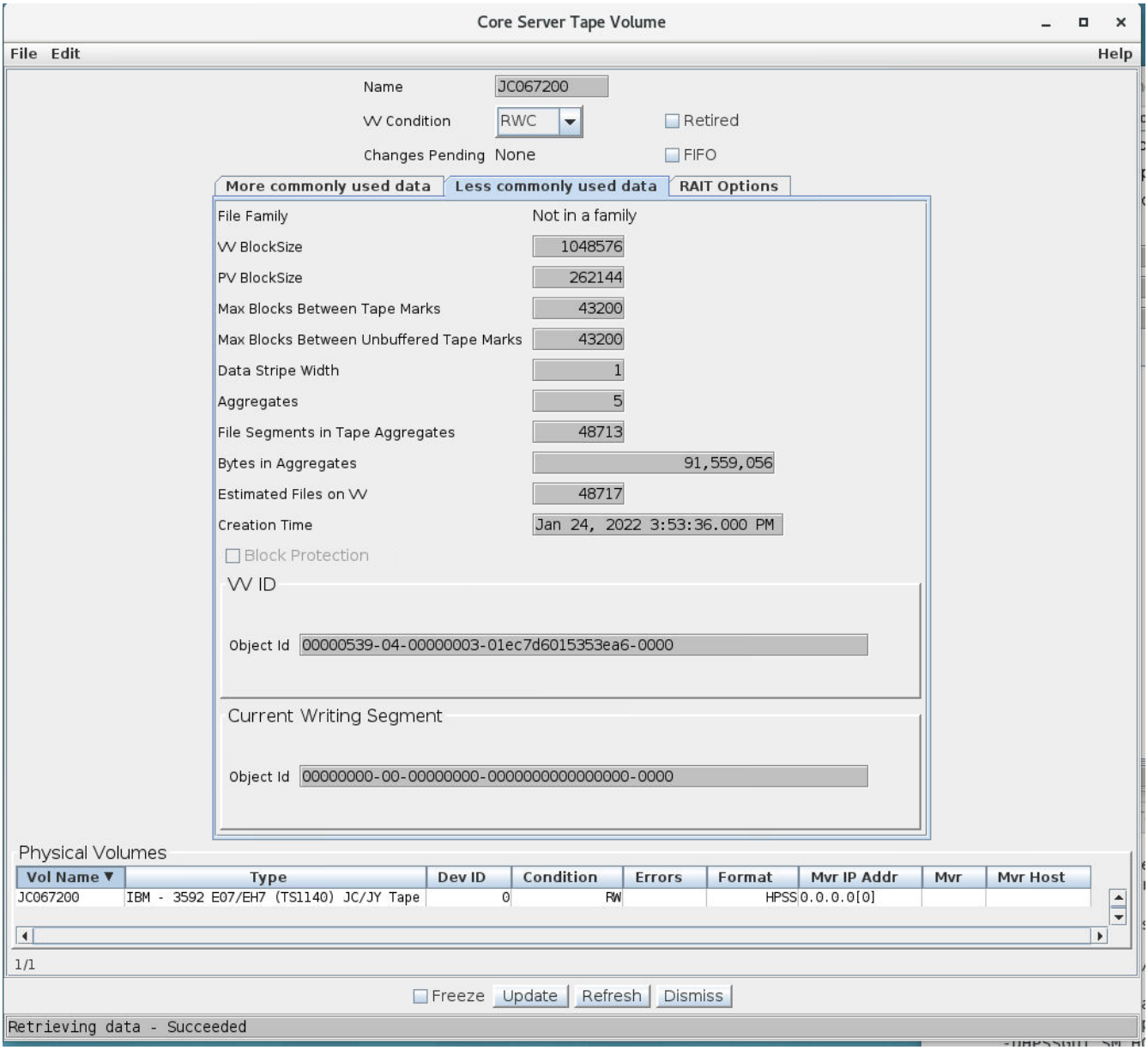
Current Position

The address to which the tape is positioned, expressed as an HPSS Relative Stripe Address. The Relative Address consists of two parts, the tape section and the byte offset. Tape sections are the data written between a pair of tape marks. The byte offset is the offset from the beginning of the current section to the place where the tape read or write is to take place.

When reading the tape, this address is set to the initial location where the tape read begins, then advances as the read progresses. When writing a tape, this value is set to the **Next Write Address**, and then advances as the tape is written. Updates to this field occur when I/O operations complete, so in some cases significant amounts of time may pass between updates.

Next Write Address

The next address that will be written on the tape volume, expressed as an HPSS Relative Stripe Address.



Less commonly used data tab

File Family

The family to which the volume is assigned, if any. If it is not assigned to a family, it is assigned to the default family, family zero.

VV Block Size

The virtual volume block size. This is the number of bytes written from a data stream to an element of the striped volume before the stream switches to the next element of the stripe.

PV Block Size

The size, in bytes, of the media data block.

Max Blocks Between Tape Marks

The maximum number of media blocks that will be written to this tape before a tape mark is written. The product of this value and the Media Block Size defines the Physical Volume Section Length.

Data Stripe Width

The number of physical volumes in the stripe group that contain file data. All volumes in HPSS are considered to be striped, even if this value is one.

Aggregates

The number of tape aggregates on this volume.

File Segments in Tape Aggregates

The number of file segments in the tape aggregates on this volume. This is a count of files within all aggregates on the volume. Files may be split across tape aggregates on the volume, which can lead them to counted multiple times in this statistic.

Bytes in Aggregates

The number of bytes stored in the tape aggregates on this volume. This value does not include parity bytes in RAIT volumes.

Estimated Files on VV

An estimate of the number of files stored on the tape volume. This estimate is arrived at by summing the number of individual tape storage segments on the volume (the server assumes one file per segment) and adding the number of files stored in aggregates on the volume. Since the number of files in an aggregate can be relatively large, the number of files on the volume may be much larger than **Number of Active Segments**.

Creation Time

The date and time the tape volume was created in the Core Server.

Block Protection

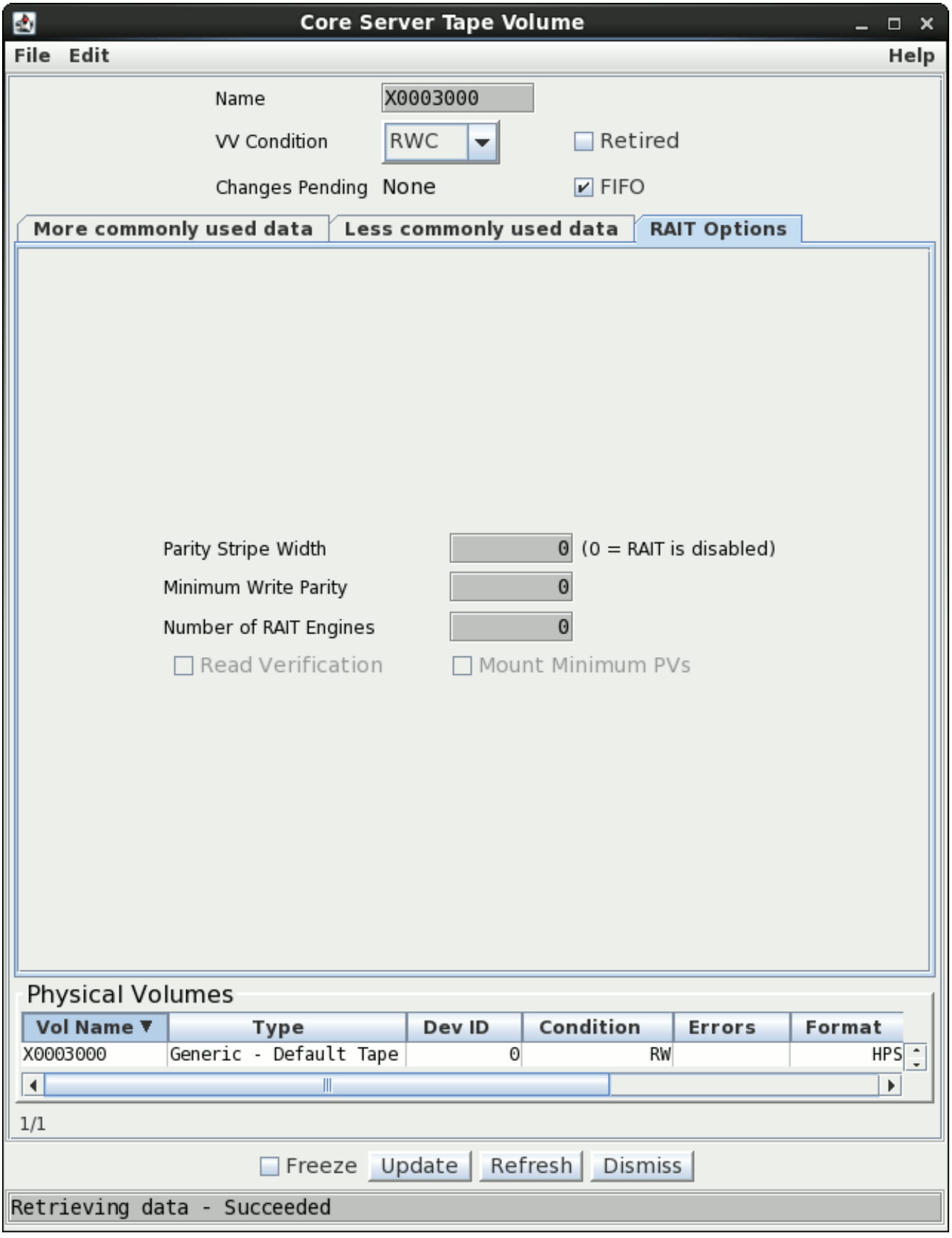
When this flag is set, Logical Block Protect (LBP) will be enabled for the physical tape volumes in this virtual tape volume. When LBP is enabled, HPSS will generate and pass a CRC with each block written to the tape volume and validate the CRC accompanying each block read from the tape volume.

VV ID

The detailed description of the tape virtual volume ID associated with this tape virtual volume.

Current Writing Segment

The detailed description of the tape storage segment ID associated with the segment that is found at the end of the tape volume. For various reasons, some tape volumes do not have a value in this field. It will be displayed filled with blanks in that case.



RAIT Options tab

Parity Stripe Width

The number of physical volumes in the tape virtual volume that contain parity information. If this value is zero, the VV is not RAIT.

Minimum Write Parity

The minimum number of parity PVs that must remain writable in order for the RAIT VV to be writable.

Number of RAIT Engines

The number or RAIT Engines that will be used when reading or writing the volume.

Read Verification

When set, parity information in the RAIT VV will be used to verify the integrity of data read from the VV. This field can be modified on this screen.

Mount Minimum PVs

When set, the smallest number of PVs necessary will be mounted when reading the RAIT VV. When **Read Verification** is not set, this will be equal to the value of DSW; and when **Read Verification** is set, this will equal (DSW + 1). This field can be modified on this screen.

Physical Volumes

This is a table of physical volume attributes for the physical volumes that make up this tape virtual volume.

Vol Name

The ASCII name of the physical volume.

Type

The media type.

Dev ID

The ID of the device the physical volume is mounted on. If the volume is not mounted, this field will be zero.

Condition

The administrative condition of the individual PV. In non-RAIT VVs, this value will always be `RW`. In RAIT VVs, this value may be `RW`, `RO`, or `DOWN` and may be changed using this screen.

Errors

The accumulated error history of the PV. In non-RAIT VVs, this field will always be blank. In RAIT VVs, this field may contain an `M` if the PV had a mount error, an `R`" if the PV had a read error, and a `"W` if the PV had a write error. The specific error history is recorded in metadata and accessible via the lspvhist(7) utility program.

Format

The recording format of the tape. Native HPSS tapes will show `HPSS` in this field. Tapes imported to HPSS from other systems will show other values.

Mvr IP Addr

The IP address of the Mover that operates this physical volume. If the volume is not mounted, this field will be zeros.

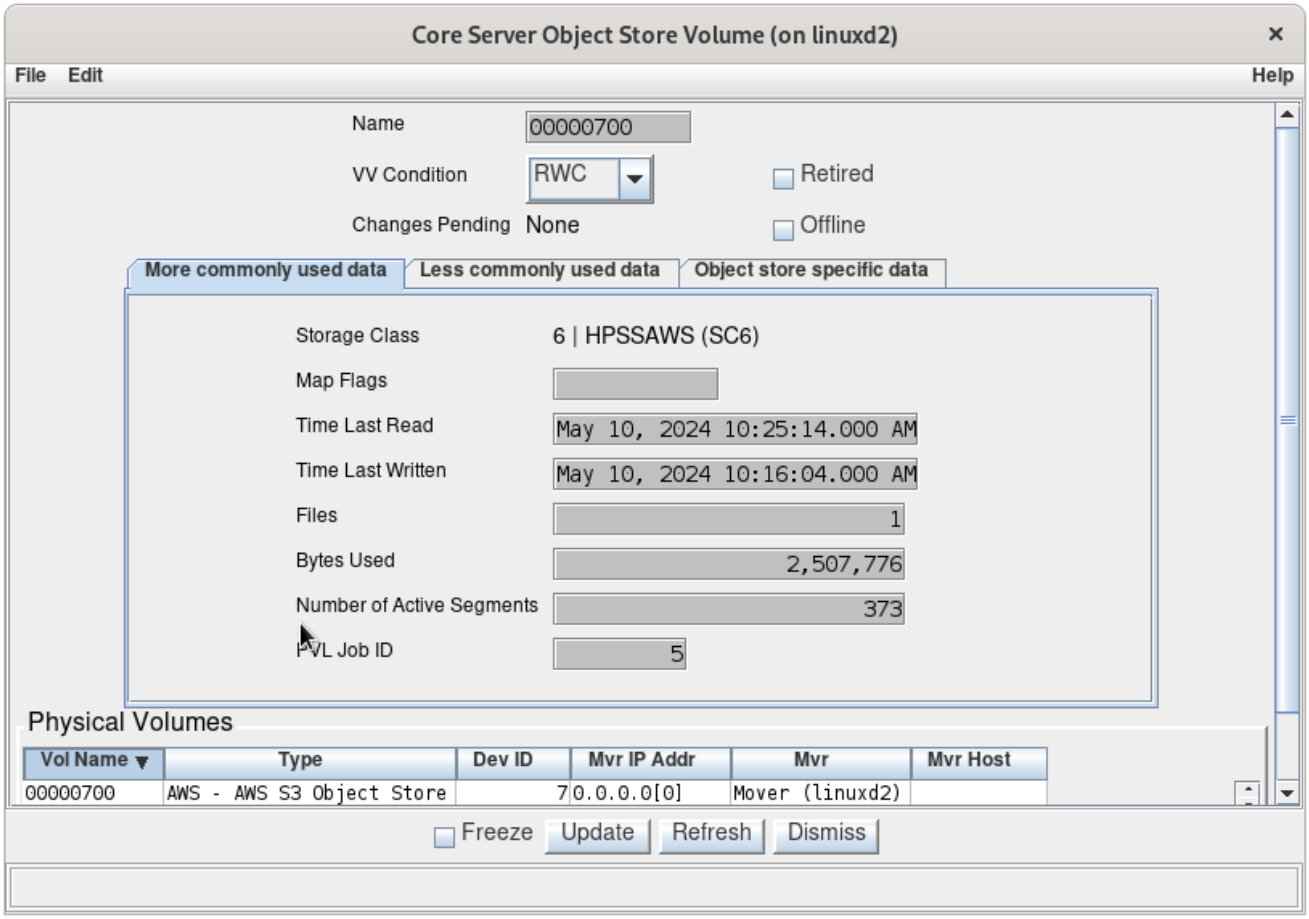
Mvr

The descriptive name of the Mover that operates this physical volume. If the volume is not mounted, this field will be blank.

Mvr Host

The name of the host on which the Mover runs. If the volume is not mounted, this field will be blank.

19.6.4.3. Core Server Object Store Volume information window



This window displays information about an object store volume as represented by the Core Server.

Field descriptions

Name

The ASCII name of the object store physical volume that corresponds to the object store virtual volume.

VV Condition

This is the administrative control for the object store virtual volume. It will have one of five values: RWC, RW, RO, EMPTY or DOWN. The RW value is not supported for object store volumes.

- In RWC condition, the volume can be read and written. This is the normal operational state.
- In RO condition, the volume can be read but not written. New storage segments cannot be created on the volume.

- In EMPTY condition, the volume cannot be read or written. All storage segments have been removed from the volume, and new segments cannot be created. Since this condition represents a physical state of the volume, the operator may not change the volume to this condition. EMPTY is displayed by the server when the condition is RO and there are no storage segments in the volume.
- In DOWN condition, the volume cannot be read, written or mounted. This condition can be used to make a object store unavailable to the system.

Change the **VV Condition** of a disk virtual volume by selecting the desired condition from the drop-down list and then clicking the **Update** button.

Changes Pending

If there are any VV Condition changes for this volume pending in the Core server, **Changes Pending** will be `Yes` with a red bullet. Otherwise, **Changes Pending** will be `None`.

Retired

When checked, the volume is retired. New storage segments will not be created on the volume.



Currently, object store volumes cannot be retired.

Offline

When checked, the volume is considered offline and not usable for reading nor writing. This often occurs as a result of failed communication with the Mover that controls the volume. HPSS will attempt to automatically return offline object stores to service. For object stores that HPSS can automatically set back online, you will see corresponding entries in the hpssgui(7) *Alarms and Events* window. Because this occurs in the background, failures to automatically set object store volumes back online are silent.

If an object store is functioning sporadically, the object store may transition back and forth between offline and online. If you want to take the object store out of service for repairs, set its VV Condition to DOWN.

A reminder about offline object stores will appear in the hpssgui(7) *Alarms and Events* window on a regular basis. By default, the reminder message appears once a day in the mid-morning. You can alter the frequency with which the message appears by setting the HPSS_OFFLINE_OBJSTOR_REMINDER_INTERVAL_HOURS environment variable. Note that the value you specify should be a number of hours, not minutes nor seconds.



Offline versus DOWN

Disks that are either offline or have a VV Condition of DOWN are unusable for read and write activity. They are essentially *out of service*. Though both states, offline and DOWN, achieve the same end result, they reflect different situations.

The offline state reflects a physical reality (for example, the object store's Mover may be down or not reachable over the network). It is automatically managed by HPSS and is not meant to be used to administratively control access to object store volumes. In general, if you want a object store to be unusable, *set its VV*

*Condition to DOWN rather than setting the **Offline** state.*

DOWN is an administrative state. It's useful for removing a object store from service regardless of whether it has an apparent malfunction.

More commonly used data tab

Storage Class

The storage class to which the object store virtual volume is assigned.

Map Flags

This field is usually blank for volumes that are actively being used to store incoming data. This field is informational only.

Time Last Read

The date and time the volume was last read. This value is not stored in metadata, but is initialized to zero when the Core Server starts and then updated each time the object store is read. This field is not affected by transaction aborts.

Time Last Written

The date and time the volume was last written. This value is not stored in metadata, but is initialized to zero when the Core Server starts and then updated each time the object store is written. This field is not affected by transaction aborts.

Files

The number of files represented by active segments allocated in this object store virtual volume.

Bytes Used

The number of bytes represented by active segment allocated in this object store virtual volume.

Number of Active Segments

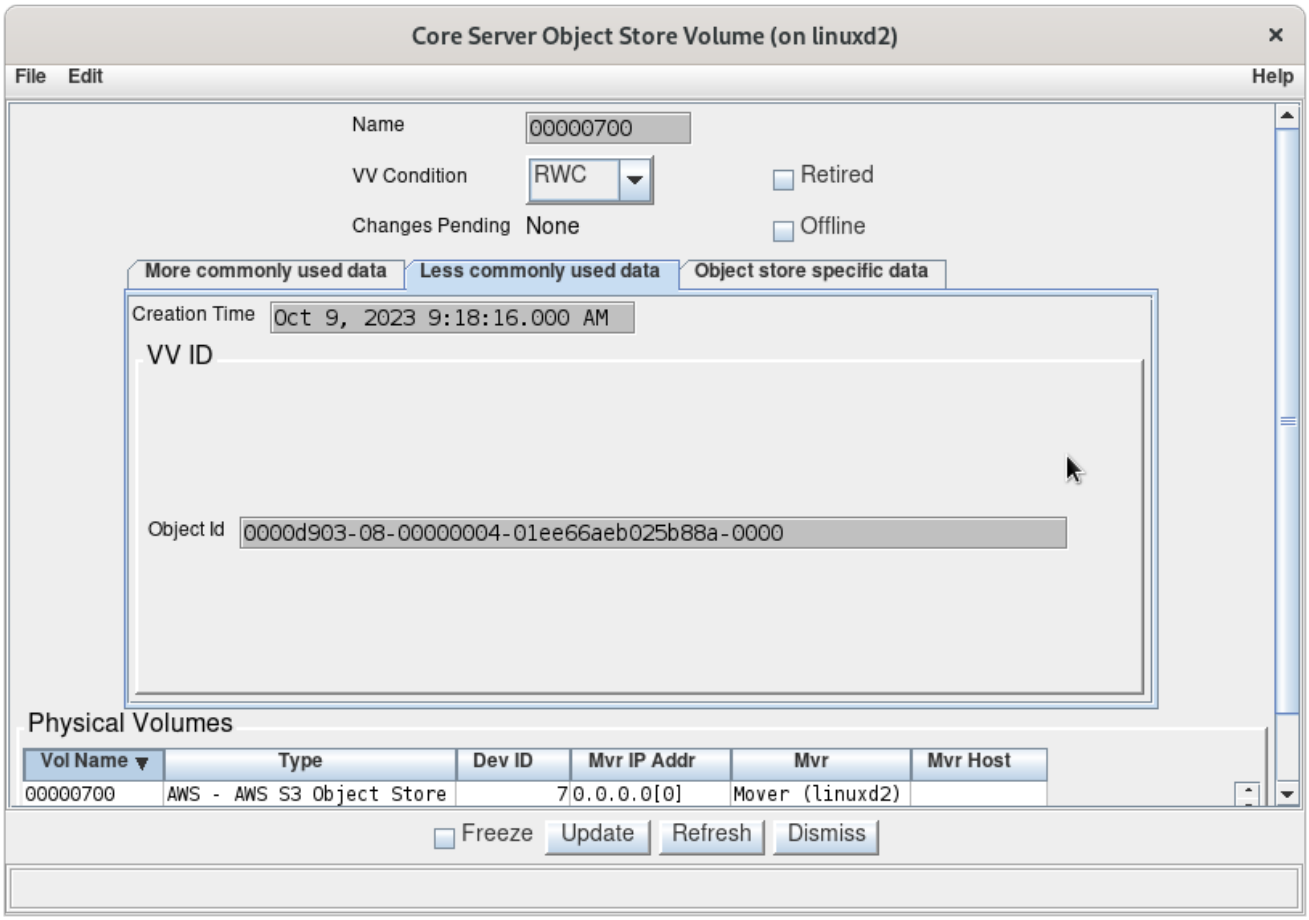
The number of object store storage segments allocated in this virtual volume.



This number should corresponding with the number of object store objects being actively referenced by segments allocated in this object store virtual volume. If object versioning is enabled for a bucket this number could be less than the actual number of objects in the objects store.

PVL Job ID

The PVL job ID in which this volume is mounted. This field will be zero if the volume is not mounted.



Less commonly used data tab

Creation Time

The date and time the object store volume was created in the Core Server.

VV ID

The detailed description of the object store virtual volume ID associated with this object store virtual volume.

Physical Volumes

This is a table of physical volume attributes for the physical volumes that make up this object store virtual volume.

Vol Name

The ASCII name of the physical volume.

Type

The media type.

Dev ID

The ID of the device the physical volume is mounted on.

Mvr IP Addr

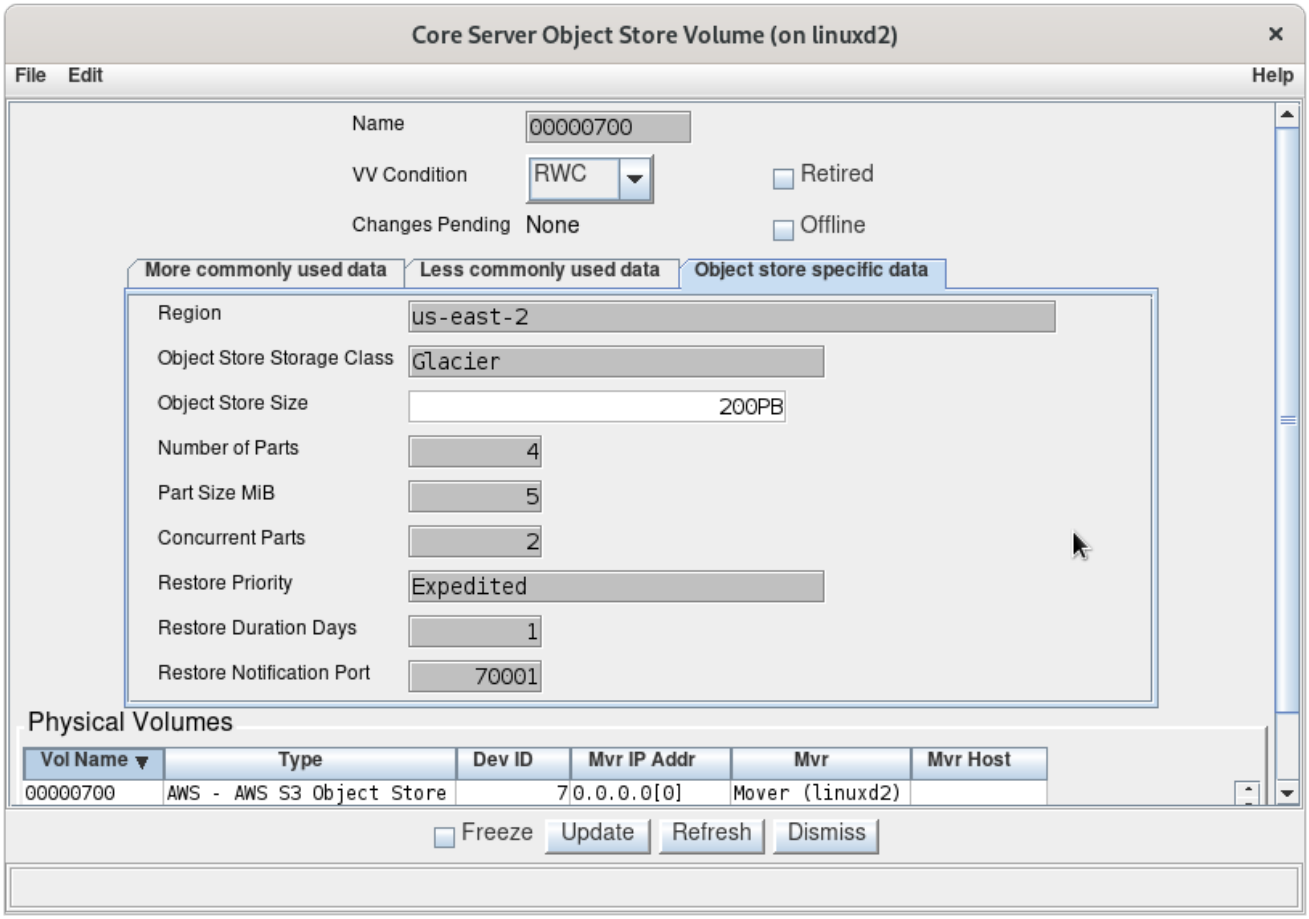
The IP address of the Mover that operates this physical volume.

Mvr

The descriptive name of the Mover that operates this physical volume.

Mvr Host

The name of the host on which the Mover runs.



Object store specific data tab

Region

The region associated with the storage class.

Object Store Storage Class

The AWS Storage Class used when storing data for this HPSS storage class.

Object Store Size

Informational limit on the number bytes to be stored in the object store via this resource.



This setting will not limit the amount of data written to the object store. It is provided only as a tool to track the current vs. expected space usage.

Number of Parts

This is the number of parts for the upload of object stored via this resource.

Part Size MiB

This is the minimum number of megabytes used for each part of a multipart upload.

Concurrent Parts

This is the number of upload parts that are transmitted concurrently.

Restore Priority

For archive type storage classes, this is the priority to be used when an object must be restored.

Restore Duration Days

The duration that restored data will remain available for recall.

Restore Notification Port

This flags indicates that the restore requests will register for notification to be sent to HPSS at completion of a restore.

19.6.5. Changing Core Server volume condition

HPSS provides a single control that sets the operational condition of a disk, tape, or object store virtual volume. The control is found on the *Core Server Tape Volume*, *Core Server Disk Volume*, and *Core Server Object Store Volume* windows and is called the VV Condition.

VV Condition controls the availability of the volume for the following actions:

- Creation of new storage segments
- Reading of existing storage segments
- Writing of existing storage segments
- Mounting of tape media

Tape volumes have five possible settings for VV Condition:

RWC

Read, Write, Create

RO

Read Only

EOM

End of Media reached

EMPTY

The volume reached EOM, and all data has been removed.

DOWN

Volume not available

Disk volumes have five possible settings for VV Condition:

RWC

Read, Write, Create

RW

Read, Write

RO

Read Only

EMPTY

The volume is in RO condition and all data has been removed.

DOWN

Volume not available

Object Store volumes have five possible settings for VV Condition:

RWC

Read, Write, Create

RO

Read Only

EMPTY

The volume is in RO condition and all data has been removed.

DOWN

Volume not available

A volume in RWC condition is fully enabled for use. This is the normal setting for all disk volumes, object store volumes, and for tape volumes that have not reached EOM. Storage segments can be created on the volume; existing segments can be read and written. Volumes in RWC condition can be changed to any other condition.

Existing storage segments on disk volumes in RW condition can be read and written, but new segments cannot be created. This setting is useful when emptying a disk through attrition. Disk volumes can be emptied more quickly by setting the condition to RW, then purging or repacking the disk. Volumes in RW condition can be changed to any other condition except as limited by the Retire flag (see below).

RO condition is similar to RW condition, but writes to existing disk storage segments are not permitted.

Storage segments on tape and object store volumes in RO condition can be read, but not written. New segments cannot be created. Tapes in RO condition are similar to tapes in EOM condition with the exception that their condition can be changed to RWC or RW while tapes in EOM condition cannot. Volumes in RO condition can be changed to any other condition except as limited by the Retire flag (see below).

Tape volumes enter EOM condition when the end of media marker is reached while writing the tape or when the administrator sets the condition to EOM. EOM condition is similar to RO condition - segments can be read, but not written. Unlike RO condition, tapes in EOM condition can only be

changed to DOWN. EOM volumes cannot enter either RWC or RO condition.

Volumes in DOWN condition cannot be read, written, created on or mounted. This setting effectively removes the volume from the system while maintaining the records of the storage segments on it and is useful for dealing with failed disk, object store, or tape volumes. Disks in DOWN condition can be changed to RWC, RW or RO condition. Tapes in DOWN condition can be changed to RWC or RO condition if they have not reached EOM. Tapes that have reached EOM can only be changed from DOWN to EOM. Object Stores in DOWN condition can be changed to RWC or RO condition.

Disk or tape volumes may be reported in EMPTY condition, but the administrator can never change a disk or tape condition to EMPTY manually. This condition will be reported when a tape in EOM condition becomes empty, or a disk in RO condition becomes empty.

The **Retire** flag controls entry into RWC condition. When the **Retire** flag is set, the condition cannot be changed to RWC, but all other changes are permitted subject to the rules outlined above. By preventing entry to RWC condition, retired volumes can be read and written, but new storage segments cannot be added to the volume. Normal attrition will gradually empty the volume.

To retire a disk or tape volume, set the condition to any setting other than RWC (typically RW for disks and RO for tapes) and then set the **Retire** flag. These two changes can be done in a single update of the *Core Server Disk Volume* or *Core Server Tape Volume* window. Note that tapes do not need to be in EOM condition to be retired.



Currently, object store volumes cannot be retired.

The **Retire** flag can be cleared if necessary. Once cleared, the condition can be changed to RWC, subject to the rules outlined above.

19.6.6. Moving PVR cartridges to another PVR

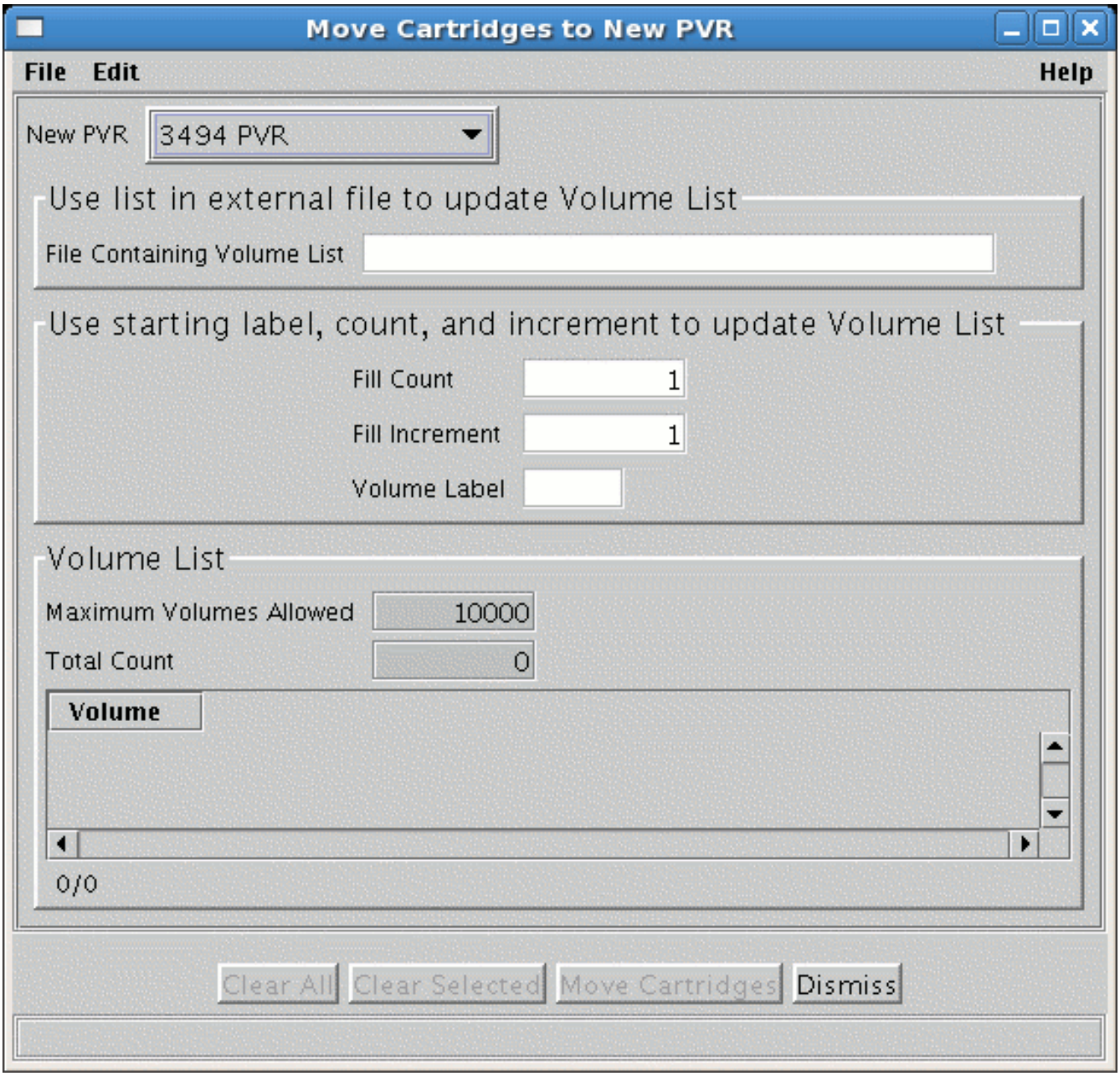
The intent of the Move Cartridge operation is to notify HPSS that a cartridge has been moved from one tape library to another.

Note that this operation will not cause a cartridge to move. Instead, the cartridge should be moved from the old library to the new library manually before informing HPSS. The operation will fail if the cartridge has not been ejected from its original robot and injected into the new robot. See the operator manuals for the specific robots involved to determine the procedures for manually ejecting and injecting a cartridge. Either the source or destination PVR (or both) during a move operation may be an operator mounted set of drives instead of a robot.

Specifically for shelf tapes, when a Move Cartridge is issued to a shelf tape, the target PVR will check whether or not the cartridge is physically in the target library. If it is not, then it will be treated as a move of a shelf tape to a shelf tape in the target library (that is, maintain its checked-out/shelf status). If it is physically in the target library, then it will be treated as a move of a shelf tape to an online cartridge in the target library.

Use the *Move Cartridges to New PVR* window in the following section.

19.6.6.1. Move Cartridges window



This window allows you to change which PVR owns a set of cartridges. Before initiating the request from the SSM window, the cartridges must already be physically placed into a tape library managed by the new PVR.

The list of the volumes to be moved may be constructed in any of three ways. Each volume name may be typed in one at a time, or a list of volume names may be automatically generated from a single entry, or a list of volume names may be specified from an input file. Volume names are not entered directly into the **Volume List** at the bottom of the window.

Any of the three entry methods may be repeated multiple times on the same window to add additional volumes to the list. All three entry methods or any combination of them may be used in succession on the same window.

To add a single volume name to the **Volume List**, set the **Fill Count** and the **Fill Increment** fields each to "1" and type the volume name into the **Volume Label** field. The volume name will be added to the **Volume List**.

To automatically generate a list of volume names in the **Volume List**, set the **Fill Count** to the desired number of volumes. Set the **Fill Increment** to the number by which each automatically generated label should differ from the previous one. Then type the starting volume name into the **Volume Label** field. The specified number of volume names will be added to the **Volume List**, each one larger than the previous entry by the specified **Fill Increment**.

Example:

```
Fill Count = 6
Fill Increment = 10
Volume Label = "AA0070"

Labels automatically inserted into Volume List:
"AA0070"
"AA0080"
"AA0090"
"AA0100"
"AA0110"
"AA0120"
```

When an addition produces overflow in a column, numerical columns are carried over properly to alphabetic columns and vice versa.

Example:

```
Fill Count = 6
Fill Increment = 2000
Volume Label= "AA7329"

Labels automatically inserted into Volume List:
"AA7329"
"AA9329"
"AB1329"
"AB3329"
"AB5329"
"AB7329"
```

The filling will not occur and an error will be displayed if the specified values would generate an invalid volume label (for example, one greater than zzz999).

To specify a list of volumes from a file, create a file containing the name of each volume to be moved on a separate line. Volume names must be six alphanumeric characters. No other characters are allowed on the line. The file must be accessible from the host on which the hpssgui is executing. Enter the name of the file in the **File Containing Volume List** field. The volume names from the file will be added to the **Volume List**.

Each cartridge in the list is sent to the System Manager as a separate move request. If an error occurs on a request, SSM stops processing the list of cartridges at that point. The window status line

will report the number of successful moves, and the cartridge name that caused the failure. For example, if you try to move a cartridge to a PVR, and that cartridge is already controlled by that PVR, an error will occur. Therefore, if a multi-cartridge move fails, you cannot click the **Move Cartridge** button to retry the same list of cartridges, because the ones that succeeded before will now fail. You must first remove the successfully moved cartridges from the list. The **Clear Selected** button is provided for this purpose.

Field descriptions

New PVR

The descriptive name of the PVR to which the cartridges will be moved.

File Containing Volume List

The name of an external file containing a list of cartridge labels to be added to the end of the **Volume List**.

Fill Count

The number of cartridge labels to be added to the end of the list when the **Volume Label** field is next modified. This number may be one or greater. If the list fills up before the **Fill Count** is exhausted, filling stops, and a message box is displayed (see **Maximum Volumes Allowed** below).

Fill Increment

In a multiple-cartridge fill, this field determines how each new cartridge label is generated from the previous one. A cartridge label is six alphanumeric characters. The **Fill Increment** is added to the least significant part of a cartridge label to generate a new label.

Volume Label

The cartridge label to be added to the end of the **Volume List**. If **Fill Count** is greater than "1", this will be the first label of a fill sequence. The label must be exactly six characters long.

Maximum Volumes Allowed

The maximum number of cartridge labels that will be allowed in the **Volume List** (10,000).

Total Count

The total number of labels currently in the **Volume List**.

Volume List

A list of cartridge/volume labels specifying the cartridges to be moved. You cannot enter labels directly into this list, but must construct it in one of the three ways described above. Use the scrollbar to move up and down the list of labels.

You can select one or more labels from the list for the purpose of removing them from the list (see **Clear Selected** below).

Buttons

Clear All

Clears the **Volume List**, and resets **Fill Count** and **Fill Increment** to "1".

Clear Selected

If one or more cartridges are selected (highlighted) in the **Volume List**, clicking on this button will remove them from the list. Note that this does not actually delete anything from HPSS.

Move Cartridges

Begins moving the listed cartridges. A start message is displayed on the message line at the bottom of the window, and all window features are disabled, except for the **Dismiss** button; this prevents you from modifying any data on the window while the move is in progress.

It is a good idea to have the HPSS *Alarms and Events* window open while moving proceeds, so that all relevant log messages can be viewed. Move operation initiation, completion and errors will be reported in the HPSS *Alarms and Events* window.

While moving proceeds, status messages are displayed on the status bar. When the move is finished, a completion message is displayed and the window features are re-enabled. At this point new data can be entered for a new move.

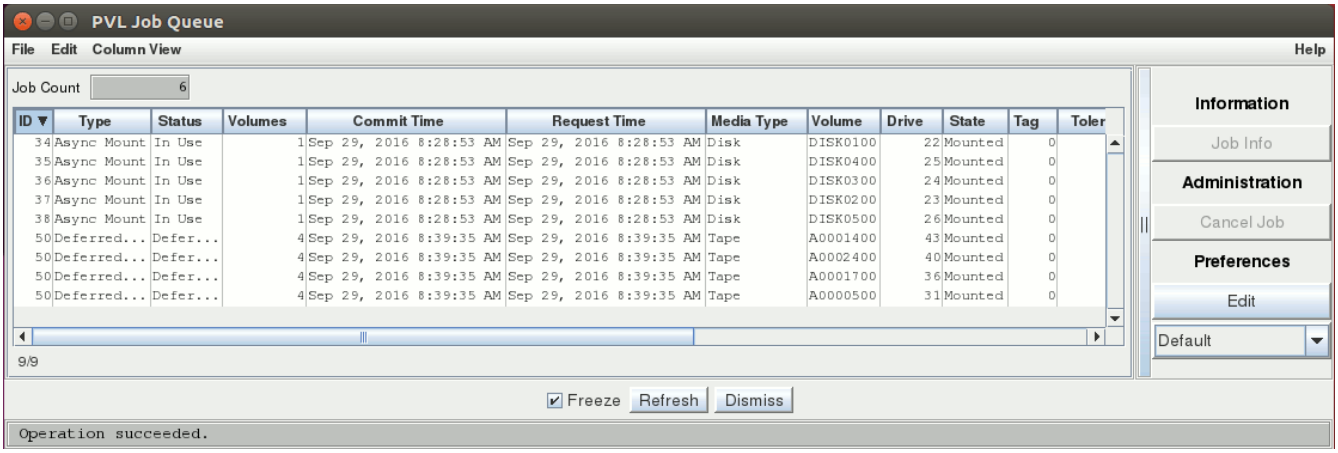
19.7. Monitoring and managing volume mounts

Volume mounts are performed by the PVL. Every volume mount is associated with a PVL job. The *PVL Job Queue* window lists all PVL jobs and volume mounts for each job. The *PVL Request Information* window lists detailed information about a single PVL job.

The PVL reports every tape mount request in the *Tape Mount Requests* window. Tapes which need to be returned to the robot from a shelf or vault location are reported in the *Tape Check-In Requests* window.

Each of these windows is described in detail in this section.

19.7.1. PVL Job Queue window



This window shows all outstanding jobs in the PVL and the mounted volume information for each job. From this window, the user can issue a request to view more information for a particular PVL job or to cancel it. Each PVL job represents a volume mount (or series of mounts for a striped disk

or tape volume).

Field descriptions

Job Count

The number of PVL jobs in the queue.

Job List

This is the main part of the window, consisting of a table of job information, a title line containing labels for each column, and vertical and horizontal scrollbars.

The function buttons to the right of the list require that a job be selected from the list. To select a job, click on it with the mouse; the selection will be highlighted.

The fields displayed in the table, as identified in the title line column headings, are shown below.

ID

A unique number assigned to each job. There may be multiple entries per job ID.

Type

The type of job. Possible types are:

- `Async Mount` indicates an asynchronous mount.
- `Default Import` indicates a media import of type "Default Import".
- `Scratch Import` indicates a media import of type "Scratch Import".
- `Overwrite Import` indicates a media import of type "Overwrite Import".
- `Export` indicates a media export.
- `Deferred Dismount` indicates a cartridge is scheduled for dismount, but the dismount has been delayed in case the cartridge needs to be used by another upcoming job.
- `Move` indicates a cartridge being moved to a new PVR.
- `Relabel` indicates a cartridge being relabeled.
- `Sync Mount` indicates a synchronous mount.
- `Tape Check-In` indicates a cartridge being added to the library.
- `Tape Check-Out` indicates a cartridge being removed from the library to be placed on the shelf or vault.
- `In Use Reduced` indicates a mount error occurred but the resulting set of mounted PV's are within the minimum number allowed for a RAIT volume.

Status

The current status of the job. Possible values are:

- `Uncommitted` indicates the mount job is a multi-part job that has been started, but the last volumes have not been added, and the mount operation has not been committed.
- `Cartridge Wait` indicates the job is waiting for another job to release a cartridge that it needs.

- `Drive Wait` indicates the job is waiting for a drive to become available.
- `Mount Wait` indicates the job is waiting for a volume to be mounted.
- `Mounted` indicates all volumes required for the job are mounted.
- `Dismount Pending` indicates volumes are in the process of being dismounted.
- `Aborting` indicates the job is being aborted.
- `Inject` indicates cartridges are being injected into a PVR (for import jobs only).
- `Eject` indicates cartridges are being ejected from a PVR (for export jobs only).
- `In Use` indicates all volumes required for the job are mounted and ready for use.
- `Deferred Dismount` indicates dismount for cartridges will be delayed.
- `Tape Check-In` indicates the job is waiting for the cartridge to be checked in.
- `Tape Check-Out` indicates the job is waiting for the cartridge to be checked out.
- `Completed` indicates the job is complete.

Volumes

The number of volumes that are mounted for this job.

Commit Time

The date and time when the job was committed. This will be the same as **Request Time** for all jobs except asynchronous mounts.

Request Time

The date and time when the job was sent to the PVL

Media Type

Type of media being used by job. `Disk`, `Tape`, or `Unknown`. This field is useful for filtering the **Job List**.

Volume

The eight-character volume label.

Drive

The numeric identifier of the drive in which the volume is mounted, or `none` if the volume does not have a drive associated with it.

State

The state of the volume within the context of this request. Possible values are:

- `Cart Assigned`
- `Cart Wait`
- `Dismount Pending`
- `Dismounted`
- `Drive Wait`

- `Eject`
- `Inject`
- `Mount Failed`
- `Mount Pending`
- `Mounted`
- `Reading Label`
- `Tape Check-In`
- `Tape Check-Out`
- `Uncommitted`
- `Unload Pending`

Tag

A value used to associate the physical volumes within a RAIT volume; that is, all tape volumes in a RAIT volume will have identical **Tag** values.

Tolerate

Specifies the number of physical tape volumes that can fail mounting within a RAIT volume (identical **Tag** values) and still be considered successfully mounted.

Library Unit ID (tape only)

Since library architecture varies, the method to associate related drives at a unit level will be determined through the use of a "unit ID" that is placed in the drive metadata. This will allow the administrator to analyze their specific library configuration and apply their strategy as they feel fit. The use of "unit ID" allows the library specifics to be abstracted from the PVL. Library details are the domain of the PVR. See **Defer Dismount Exempt Count** information in SCSI PVR-specific configuration window.

Drive Type

The HPSS drive type assigned to the drive.

Mover

The name of the Mover that owns the device where the volume is mounted. This field will be blank if the volume is not currently mounted.

PVR

The name of the PVR that owns the mounted volume. This field will be blank if the volume is not a tape volume.

Buttons

Job Info

Click on this button to open the *PVL Request Information* window for the selected job.

Cancel Job

Click on this button to cancel the selected job. You are prompted to confirm your request before

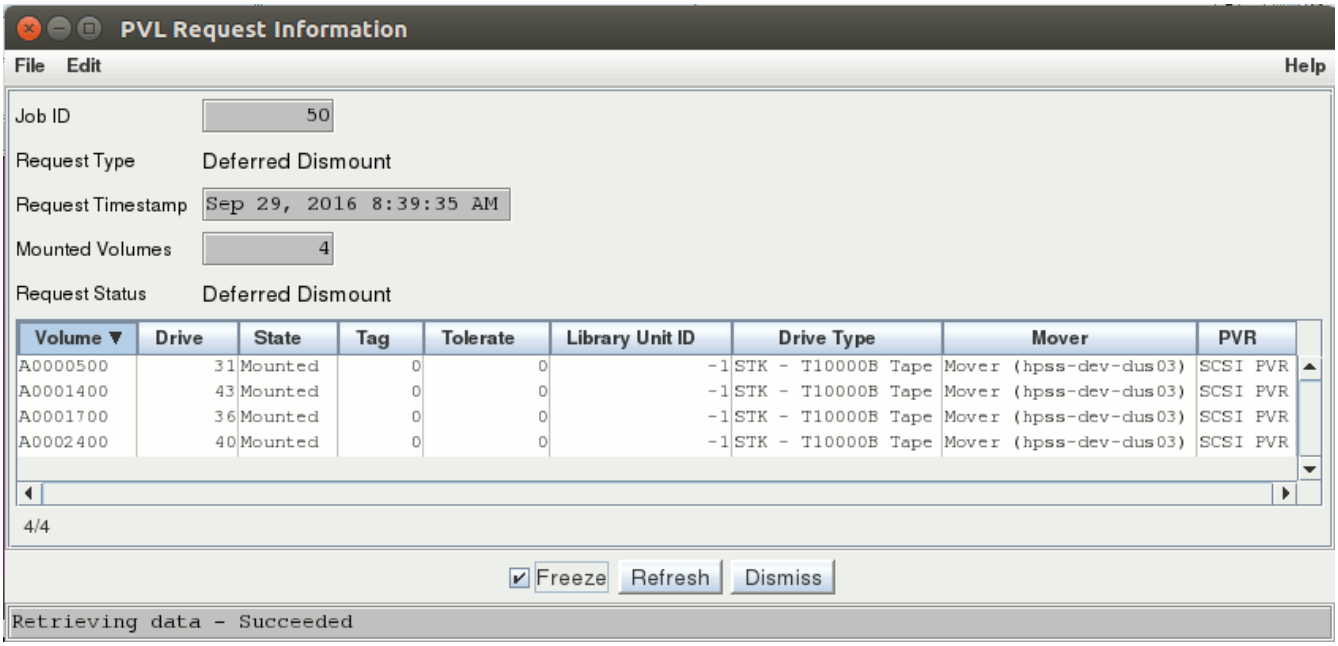
it is actually executed. This will generate a dismount request.

Related Information

Tape Mount Requests window

Canceling queued PVL requests

19.7.2. PVL Request Information window



This window is displayed when the **Job Info** button is clicked on the *PVL Job Queue* window. It allows you to view the information associated with a PVL job/request.

Field descriptions

Job ID

The unique number assigned to the job being viewed.

Request Type

The type of job/request. Possible types are:

- `Async Mount` indicates an asynchronous mount.
- `Default Import` indicates a media import of type "Default Import".
- `Scratch Import` indicates a media import of type "Scratch Import".
- `Overwrite Import` indicates a media import of type "Overwrite Import".
- `Export` indicates a media export.
- `Move` indicates a cartridge being moved to a new PVR.
- `Relabel` indicates a cartridge being relabeled.
- `Sync Mount` indicates a synchronous mount.

- `Deferred Dismount` indicates dismount delayed.
- `Tape Check-In` indicates a cartridge being added to the library.
- `Tape Check-Out` indicates a cartridge being removed from the library.
- `In Use Reduced` indicates a mount error occurred but the resulting set of mounted PVs are within the minimum number allowed for a RAIT volume.

Request Timestamp

The date and time when the request was issued to the PVL.

Mounted Volumes

The number of volumes that are currently mounted for this request.

Request Status

The current status of the job/request. Possible values are:

- `Aborting` indicates the job is being aborted.
- `Cartridge Wait` indicates the job is waiting for another job to release a cartridge that it needs.
- `Completed` indicates the job is completed. Once a job is completed, it no longer exists in the PVL job queue, and this window will no longer receive any updates.
- `Deferred Dismount` indicates dismount for cartridges will be delayed.
- `Dismount Pending` indicates volumes are in the process of being dismounted.
- `Drive Wait` indicates the job is waiting for a drive to become available.
- `Eject` indicates cartridges are being ejected from a PVR (for export jobs only).
- `In Use` indicates all volumes required for the job are mounted and ready for use.
- `Inject` indicates cartridges are being injected into a PVR (for import jobs only).
- `Mount Wait` indicates the job is waiting for a volume to be mounted.
- `Mounted` indicates all volumes required for the job are mounted.
- `Tape Check-In` indicates the job is waiting for the cartridge to be checked in.
- `Tape Check-Out` indicates the job is waiting for the cartridge to be checked out.
- `Uncommitted` indicates the job has not yet been committed (used for asynchronous mount jobs only).

Volume List

This group of fields shows information on the volumes associated with this request.

Volume

The eight-character volume label.

Drive

The numeric identifier of the drive in which the volume is mounted, or `none` if the volume does not have a drive associated with it.

State

The state of the volume within the context of this request. Possible values are:

- `Cart Assigned`
- `Cart Wait`
- `Dismount Pending`
- `Dismounted`
- `Drive Wait`
- `Eject`
- `Inject`
- `Mount Failed`
- `Mount Pending`
- `Mounted`
- `Reading Label`
- `Tape Check-In`
- `Tape Check-Out`
- `Uncommitted`
- `Unload Pending`

Tag

A value used to associate the physical volumes within a RAIT volume; that is, all tape volumes in a RAIT volume will have identical **Tag** values.

Tolerate

Specifies the number of physical tape volumes that can fail mounting within a RAIT volume (identical **Tag** values) and still be considered successfully mounted.

Library Unit ID (tape only)

Since library architecture varies, the method to associate related drives at a unit level will be determined through the use of a "unit ID" that is placed in the drive metadata. This will allow the administrator to analyze their specific library configuration and apply their strategy as they feel fit. The use of "unit ID" allows the library specifics to be abstracted from the PVL. Library details are the domain of the PVR. See **Defer Dismount Exempt Count** information in SCSI PVR-specific configuration window.

Drive Type

The HPSS drive type assigned to the drive.

Mover

The name of the Mover that owns the device where the volume is mounted. This field will be blank if the volume is not currently mounted.

PVR

The name of the PVR that owns the mounted volume. This field will be blank if the volume is not a tape volume.

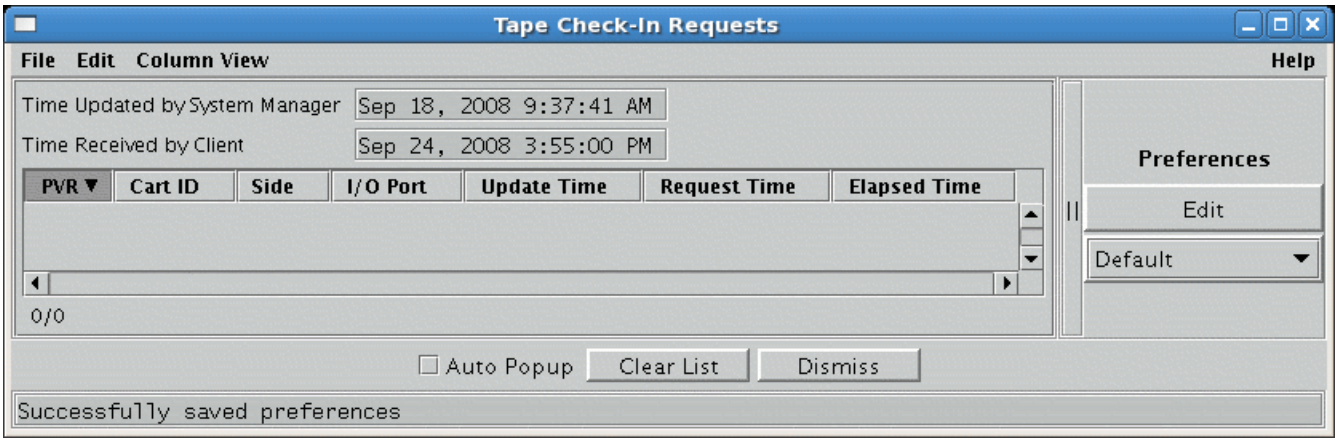
19.7.3. Canceling queued PVL requests

PVL requests that cannot be completed due to a hardware or software problem that HPSS cannot handle can be canceled by the system administrator.

Canceling a PVL job will result in the PVL issuing a dismount job request if any of the volumes in the job were mounted. The dismount request causes an unload request to be sent to the controlling Mover. This request may fail if the device is performing an I/O transfer. The PVL will retry the unload command until the I/O completes and the device reservation is freed. Once a volume has been successfully unloaded, all I/O operations directed to the device will fail. Cancelation should therefore be done only after careful consideration of its consequences.

When a deferred dismount job is canceled, the PVL issues dismount requests to all drives loaded with cartridges in deferred dismount state. Canceling a deferred dismount job can be done as often as necessary and does not result in any errors elsewhere in the system.

19.7.4. Tape Check-In Requests window



When using the shelf-tape feature of HPSS, operators are notified via the *Tape Check-In Requests* window of mount requests for tapes which are stored outside the robot (such as in a vault). Once the requested tapes are inserted back into the robot, tape check-in requests are removed from the list. If a Tape Check-In Request has been displayed, and the corresponding tape has not been checked in, the PVR will begin logging alarms indicating that the appropriate tape has yet to be checked in. The frequency of these alarms is controlled by the Shelf Tape Check-In Alarm field of the PVR-specific configuration window.

The requests are displayed in chronological order. Each time such a request is received by SSM, it is added to the list, but duplicate requests for the same tape are not displayed. When a message is received indicating that the check-in request has been satisfied, the request is removed from the window. For operator PVRs such check-in requests mean that a tape must be mounted by hand.

The maximum number of tape check-in messages that can be displayed on the *Tape Check-In Requests* window is 100. When this limit is exceeded, some tape check-in notifications will be lost.

If the **Auto Popup** check box is selected, this window will automatically reopen whenever a new tape check-in request is received. To open this window manually, select the **Monitor** menu on the *HPSS Health and Status* window, from there select the **Tape Requests** submenu, and from there select the **Check-In** menu item.

The HPSS shelf_tape(7) utility may be used to move tapes from offline storage to a tape library.

Field descriptions

Tape Check-In Requests list

PVR

The descriptive name of the PVR that owns the cartridge.

Cart ID

The six-character label identifying the cartridge to be inserted into the robot.

Side

The side (partition) number to be mounted. This may be blank.

I/O Port

The descriptive name of the I/O port where the cartridge will be placed.

Update Time

The date and time when the check-in request was last updated.

Request Time

The date and time that the check-in request was first received by the SSM.

Elapsed Time

The length of time (days-hours:min:sec) that the check-in request has been waiting.

Buttons and check boxes

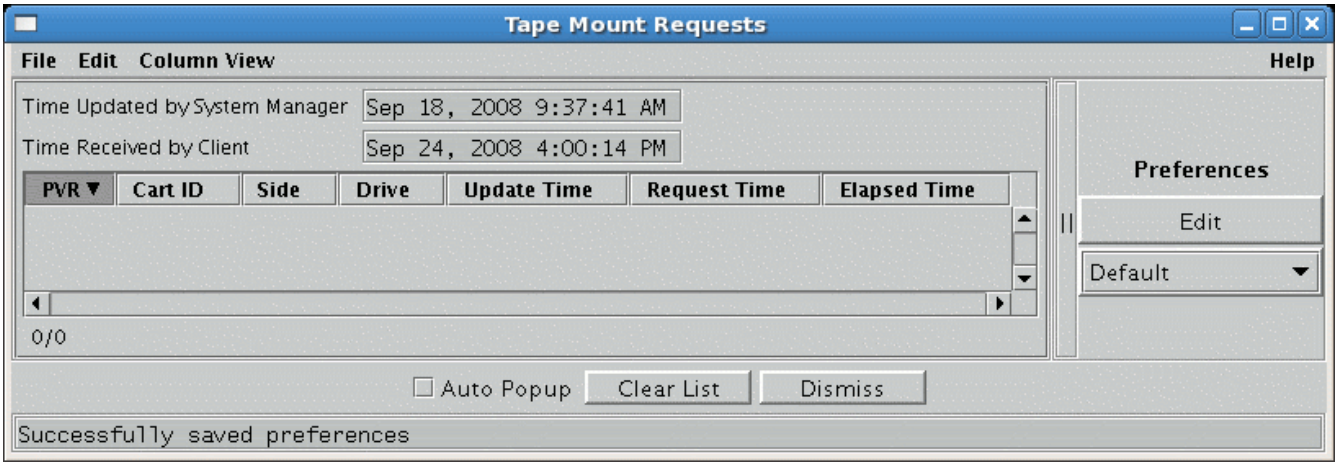
Auto Popup

If ON, this window will be opened automatically if it is not already on the screen when a check-in request is received.

Clear List

Clears the list of check-in requests. Note that this does not cancel any check-in requests, but just removes them from the list. Pending check-in requests will reappear in the window as the PVR periodically retries the check-in of cartridges. This can be useful for removing stale check-in requests that, for some reason, never issued a completion message to SSM. When this button is clicked from any SSM session, the *Tape Check-In Requests* windows on all SSM sessions will be cleared.

19.7.5. Tape Mount Requests window



The *Tape Mount Requests* window displays tapes which need to be mounted in a drive. All HPSS tape mount requests, including both robotic and operator tape mounts, will be displayed in the window. For operator PVRs, such mount requests mean that a tape must be mounted by hand. When mount requests for robotic PVRs do not disappear from the window in a timely manner, it can be an indication of hardware or other problem in the robot.

Outstanding tape mounts, along with the associated PVR, are displayed in chronological order. Each time a tape mount request is received by SSM, it is added to the list, but duplicate requests for the same cartridge are not displayed. When a message is received indicating that the mount request has been satisfied, the request is removed from the window.

The window may display an old mount request at times, due to lost notifications. Click on the **Clear List** button to refresh the window.

When a new mount notification is received, SSM may bring up the *Tape Mount Requests* window automatically if it is not currently displayed. This behavior is disabled by deselecting the **Auto Popup** check box. To open the window manually, select the **Monitor** menu on the *HPSS Health and Status* window, from there select the **Tape Requests** submenu, and from there select the **Mount** menu item.

Field descriptions

Tape Mount Requests list

PVR

The descriptive name of the PVR that owns the cartridge.

Cart ID

The six-character label identifying the cartridge to be mounted.

Side

The side (partition) number to be mounted. This may be blank.

Drive

The drive address if a specific drive was requested for the mount. In most cases, a drive is not specified, and this is blank.

Update Time

The date and time when the mount request was last updated.

Request Time

The date and time that the mount request was first received by the SSM.

Elapsed Time

The length of time (days-hours:min:sec) that the mount request has been waiting.

Buttons and check boxes

Auto Popup

If ON, this window will be opened automatically if it is not already on the screen when a mount request is received.

Clear List

Clears the list of mount requests. Note that this does not cancel any mount requests, but just removes them from the list. Pending mounts will reappear in the window as the PVR periodically retries the mounts. This can be useful for removing stale mount requests that, for some reason, never issued a completion message to SSM. When this button is clicked from any SSM session, the *Tape Mount Requests* windows on all SSM sessions will be cleared.

19.7.6. Administrative tape dismounts

HPSS provides the ability to administratively command a tape dismount in unusual situations, such as when an administrator wants a tape dismounted without waiting for HPSS to do so, or when HPSS is unable to do so due to a failure.

Administrative dismounts should only occur in exceptional error circumstances. Dismounting a tape may cause the operations being performed on that tape to fail. A library dismount operation is unlikely to occur while a Mover has the device reserved for I/O.

Use the *Devices and Drives* list window to issue a dismount for the desired device/drive entry. A confirmation window will pop up requiring you to confirm the dismount request.

19.8. Storage technology replacement

As next generation storage technology becomes available, the need arises to replace prior generation storage media. The procedure for effecting this technology change involves updating a storage class's definition with the new media type and characteristics, importing new media, creating new virtual volumes (VVs), retiring prior generation VVs, moving data from old media to new media, and finally removing the old media altogether from HPSS.

The act of retiring a VV indicates to HPSS that no more data should be written to the volume. Furthermore, it indicates that the volume should not be reclaimed for future reuse. A retired volume is considered end of life, only to be used for light reading until all the data is moved off the volume. Note that a retired VV cannot have a **VV Condition** of `RWC`.

To replace prior generation media with new, the following procedure serves as a guide. HPSS administrators should be aware that the procedure will probably need to be tailored to each customer's specific circumstances and needs. This procedure applies to both tape and disk volumes.

1. Change the storage class definition by changing its media type and any other related fields.
   - The new storage class values must reflect the characteristics of the new media.
   - In addition, the new values should be compatible with other storage levels in the hierarchy.

   *See Changing a storage class definition for guidance on changing a storage class's **Media Type** and other values.*

   Changing a storage class's configuration does not apply to existent HPSS virtual volumes; a VV's storage class configuration is set upon resource creation and remains immutable for the lifetime of the VV. A new storage class configuration will only apply to new VVs that are created after modifying the storage class.

   If upgrading to a new subtype of the same tape type, setting HPSS environment variable HPSS_CORE_CREATE_ALLOW_SUBTYPE to "true" will allow you to import the new tape subtypes without first changing the storage class configuration. This is especially useful when creating resources of multiple different subtypes at once. Regardless, the storage class configuration should still be changed to the correct subtype for future tape VV creation. This environment variable is only provided to ease the transition.

2. Click **Apply Config** on the *HPSS Health and Status* screen for the modifications to the storage class to take effect.
3. Import and create new volumes.
   - See Adding storage space for details.
   - The new volumes will take on the characteristics of the redefined storage class.
   - Consider creating only one volume at first in order to check its characteristics. If they are not as desired, delete the volume and repeat the first two steps in this procedure.

   It isn't necessary to create all the new volumes at once; only as many as are required to meet the immediate demands of the system need to be created. Keep in mind, that when storage resources are created they are immediately available for use by HPSS.

4. Use the retire(7) utility program to retire the prior old volumes.

   dump_sspvs(7) can be used to create a list of physical volumes for retire(7) to process. Alternatively, if the number of volumes to retire is small, the hpssgui(7) may be used to retire them one-by-one. Do so by changing each volumes' **VV Condition** and selecting the *Retire* check box in the *Core Server*

> *Tape Volume* or *Core Server Disk Volume* window. It is not necessary to change the **VV Condition** of tape volumes to `EOM`; the `RO` VV Condition will prevent the tapes from being written and is reversible (unlike `EOM`).

5. Use the repack(7) utility program to move data from all retired volumes in the storage class.
   - This step is frequently a long process and may require regular attention from the HPSS administrators.
   - Use dump_sspvs(7) to monitor the repack(7) progress.
   - View the repack(7) man page for details on how to use the program.
6. Use the remove(7) utility program to remove all retired and empty volumes.
   - This program will delete all resources associated with these volumes and export them from HPSS.
   - View the remove(7) man page for details on how to use the program.

Chapter 20. Logging and status

20.1. Logging overview

The purpose of logging is to record events of interest that occur in HPSS in the sequence they occur to support diagnostic research.

HPSS provides eight log message types:

- Alarm
- Event
- Info
- Debug
- Request
- Security
- Accounting
- Trace

The purpose of each of these log message types is described later in this chapter.

The HPSS logging system includes these components:

- Log policies Log policies
- Syslog Managing syslog
- Log archiving Managing log archiving
- The SSM *Alarms and Events* window Alarms and Events window

A standard configuration for logging services is usually set by the administrator during the HPSS system configuration. Specialized configurations can be set up and used to provide more or less logging for site-specific or shift-specific operational situations. Increasing the level of logging may slow down overall system operation due to the overhead of extra messages, and decreasing the amount of logging may eliminate certain log messages that could prove useful in a postmortem analysis of problem situations. The logging configuration can be tailored, as needed, to satisfy site or shift requirements.

Each SSM graphical user session (hpssgui(7)) and command line session (hpssadm(7)) can write a session log file of the SSM error and informational messages issued during the session. These are not the same as the alarm, event, and other HPSS log messages described above and are not described in this chapter. See the **-S** option in the hpssgui(7) and hpssadm(7) man pages for details.

Log messages may be deposited in three places. Certain high-priority messages are transmitted to SSM and displayed on the *Alarms and Events* window. This allows the administrator to see events unfolding in close to real time. Log messages are written to the syslog subsystem. The administrator

may filter, split, or configure the behavior of the HPSS logs using their system's syslog. HPSS log messages are logged using the "HPSS" identifier. Additionally, if the HPSSLOG_CONSOLE environment variable is enabled, log messages are also reflected to stderr.

Logging is a component which is utilized by HPSS servers and utilities. The logging component reads its log policy and then filters messages based on the configured log policies. Messages which pass the filter are logged to syslog, and may also be forwarded to the *Alarms and Events* SSM window.

For useful logging to take place, it is important that log policies and syslog are all configured and managed properly.

20.2. Log policies

For any HPSS server or utility which uses the HPSS logging facility, a log policy can be configured to specify which log message types will be sent to syslog.



If no log policy is configured for a server, the server will use the default log policy.

After changing any log policy information, including the Default Log Policy, changes will be reflected the next time the logging component rereads the logging policy. Processes using the HPSS logging component automatically reread their logging policy every thirty seconds.

20.2.1. Creating a log policy

A new log policy can be created and managed using the *Logging Policy* window (see Logging Policy configuration window).

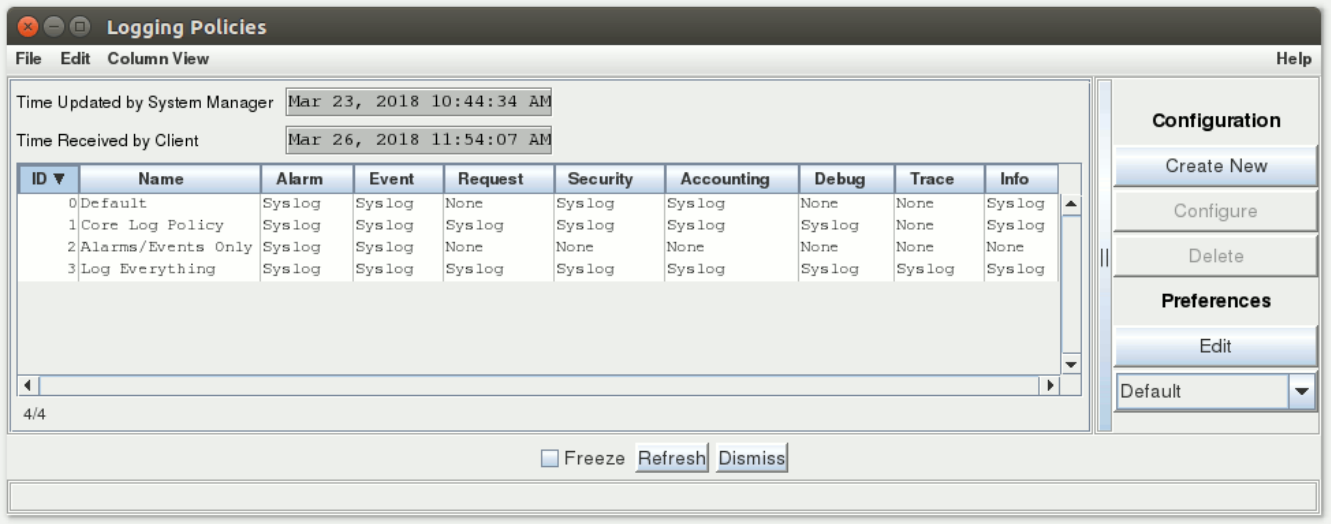
This *Logging Policy* window can be accessed from the *Logging Policies* window (see Logging Policies window).

From this window, a log policy can be created with any unique name which is typed into the **Descriptive Name** field on the *Logging Policy* window.

Newly-created servers will use the default log policy. They can be assigned a different log policy in the Server configuration.

window.

20.2.2. Logging Policies window



This window is used to manage all the log policies in the HPSS system.

To create a new log policy, click on the **Create New** button. To configure an existing log policy, highlight the policy in the list and click on the **Configure** button. When creating or configuring a log policy, the *Logging Policy* window will appear. To delete an existing log policy, highlight the policy in the list and click on the **Delete** button. If any servers still use the log policy, you will not be able to delete it. Note that the default log policy cannot be deleted and the **Delete** button is grayed out when selecting that server.

Field descriptions

Logging Policy List

Name

The descriptive name of the log policy.

Log Policy List table columns

For details on the fields listed for each log policy, refer to the *Logging Policy* window (see Logging Policy configuration window).

Configuration buttons

Create New

Opens a *Logging Policy* window containing default values for a new policy.

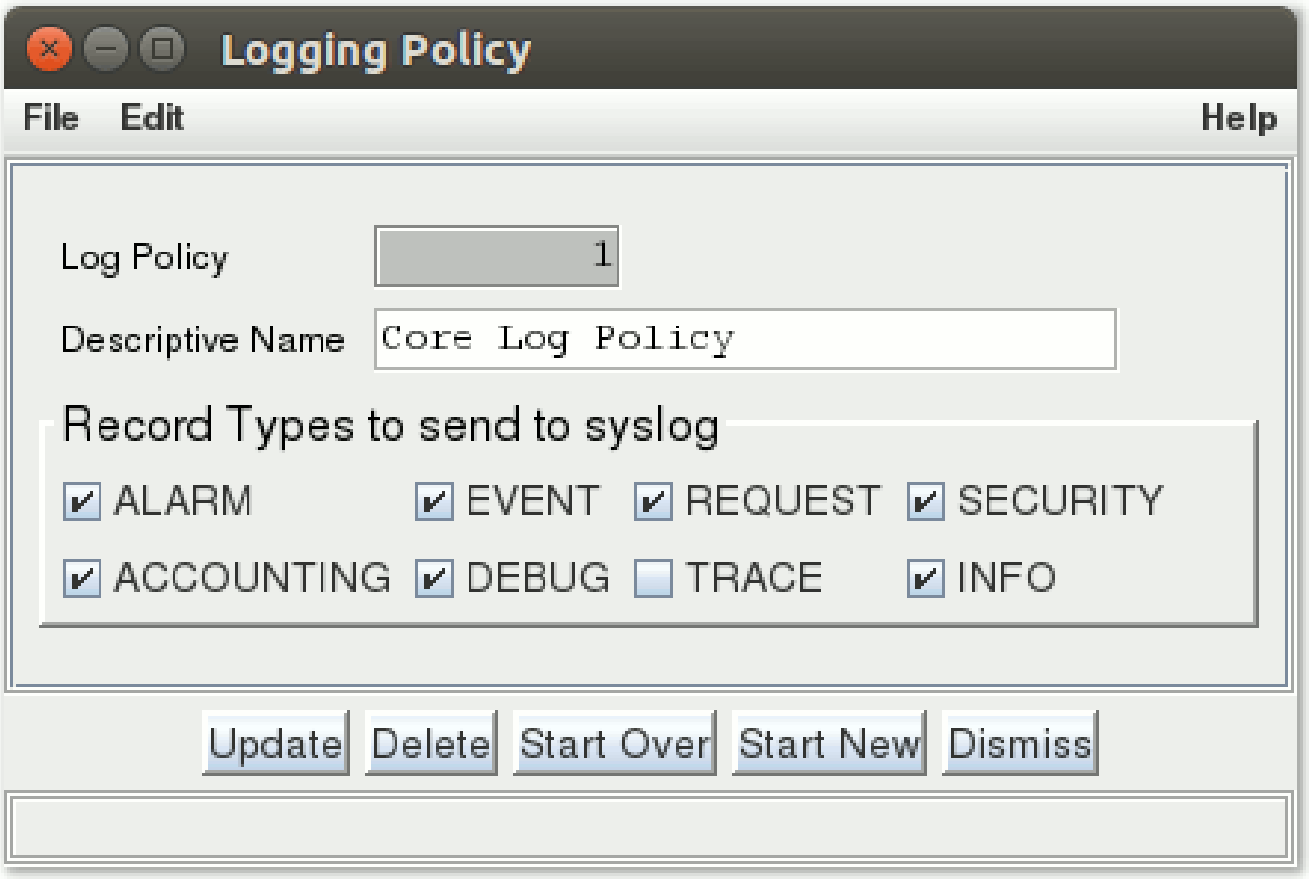
Configure

Opens the selected log policies for configuration editing.

Delete

Deletes the selected log policies.

20.2.2.1. Logging Policy configuration window



The *Logging Policy* window is used to manage a log policy.

When creating a new log policy, the **Descriptive Name** field will be set to the form "Log Policy #", where # matches the ID number for the new policy. A set of record types to send to syslog matching the default log policy will be selected.

For logging of a record type to be enabled, the check box located to the left of the log record type must be selected. The log state for each record type can be toggled by clicking on the check box.

After creating or modifying a logging policy, including the default, it may take up to thirty seconds for changes to propagate to the entire system. For example, if you modified log policy A, any servers using that policy would begin logging according to that policy within thirty seconds.

Field descriptions

Log Policy

Log policy identifier

Descriptive Name

The descriptive name of the HPSS server or utility to which this log policy applies.

Record Types to send to syslog

Record types that are to be sent to syslog for the specified server. If a check box is selected for a record type, that type will be sent to syslog. The record types are:

ALARM

A message reporting a situation that requires administrator intervention. It is recommended that this always be selected.

EVENT

An informational message (for example, subsystem initializing or subsystem terminating) about a significant occurrence in the system that is usually not an error. It is recommended that this always be selected.

REQUEST

A message which reports the beginning and ending of processing of a client request in a server. It is recommended that this type not be selected except for short periods as an aid to isolating a problem.

SECURITY

A message reporting security-related events (for example, authorization failures). It is recommended that this always be selected.

ACCOUNTING

A message which logs accounting information.

DEBUG

A message reporting an internal system event that can aid in troubleshooting problems. For troubleshooting, these messages are important record types because they provide more detailed information about the root cause of an error. It is recommended that this always be selected.

TRACE

A message reporting procedure entry and exit events. It is recommended that this type not be selected except for short periods as an aid to isolating a problem.

INFO

A message reporting progress and status for more common, longer-running requests, as opposed to certain less frequent, more significant operations covered by EVENT log messages.



It is recommended that at least the Alarm, Event, Security and Info record types be selected for all servers while they are running normally. Core Servers and Movers should also log Debug messages, and additional record types for debugging purposes should be selected for servers experiencing problems. If system performance is being degraded by excessive logging, first turn off Trace, Debug and Request record types for all servers. Other record types can be turned off as necessary; however, the HPSS administrator may lose useful debugging information when problems occur.

To facilitate diagnosis of data transfer problems, it is recommended that Debug logging always be turned on for all Movers.

20.2.3. Changing a log policy

A server's log policy can be modified to control the volume of messages to the chosen logging destinations. Typically, during normal operations, the level of logging may be decreased to only Alarm, Event, and Security to reduce overhead; however, while tracking an HPSS problem, it may be desirable to include additional log message types such as Debug, Request and Trace to obtain more information. As stated previously, it is recommended that Trace not be selected except for short periods as an aid to isolating a problem.

After a log policy has been modified, any servers using it will have their logging behavior changed within thirty seconds.

20.2.4. Deleting a log policy

There are two ways a log policy can be deleted:

- Highlight the log policy in the *Logging Policies* window (see Logging Policies window) and click the **Delete** button.
- Open the *Logging Policy* window (see Logging Policy configuration window) for the appropriate log policy and click the **Delete** button.

You will not be permitted to delete a log policy if any servers are configured to use it. Also, the default log policy cannot be deleted.

After deleting a log policy, any changes to logging, for example, reverting to use the default log policy, will begin within thirty seconds.

20.3. Managing syslog

This section describes configuring and viewing HPSS system logs. Note that for servers and processes using the logging subsystem, logs will appear in a location defined by that system's syslog configuration. HPSS log messages are text-based and may be manipulated and viewed with standard UNIX tools.

20.3.1. Configuring syslog

This documentation will assume rsyslog is being used by default. Notes and information for other syslog implementations may be found inline where there are interesting differences between implementations.



Configuring logging requires careful consideration of the site's needs and resources. Too much logging can exhaust disk space, hide relevant information, and degrade performance. Too little logging won't provide enough information for resolving system problems when they arise. Use the options in your logging policies to adjust the volume of logging information recorded in a way that is commensurate with your site's needs and resources.

HPSS log messages are sent to syslog using the "HPSS" identifier. Any file which contains HPSS

system logs is referred to as an "HPSS log file". The syslog configuration has full control over the file or files that HPSS log messages appear in, and system utilities such as **logrotate** control the log retention and log archival strategy.



When using **vim** to view HPSS log files, `:set syntax=messages` will turn on syntax highlighting similar to `/var/log/messages`.

20.3.2. Recommendations for syslog configuration

The rsyslog service has a wide variety of configuration options which are largely outside the scope of this document; however, a few items may be useful in initially setting up syslog.

With no additional configuration, HPSS logs will go into the default syslog location, which is likely `/var/log/messages`. By default, SSM will look for logs in `/var/hpss/log/HPSS.log`.



You should run the `configure_logging` tool during installation (see Configure Logging) to properly configure rsyslog before starting SSM for the first time. If you skip this step, SSM will still function but will not be able to display logs in the Alarms and Events window unless you manually configure syslog or update the SSM configuration to point to an alternative log location.

In addition to the dedicated HPSS log file, mirroring HPSS alarms and events into `/var/log/messages` may also be desirable. The recommended syslog configuration for the HPSS Core Server is located in: `$HPSS_ROOT/config/templates/system_files/hpss.conf.rsyslog.template`

With the template configuration, the Core Server will act as a syslog server where all HPSS messages will go (in `/var/hpss/log/HPSS.log`). Alarms and events will also go to `/var/log/messages`. Remember that HPSS filters logging by server based upon that server's log policy. The default port is 514.

Remote systems must also configure a syslog configuration. This causes the remote system to log back to the HPSS Core Server. The remote template needs to be updated to include your HPSS Core Server in the forwarding line. The recommended syslog configuration for the HPSS Core Server is located in: `$HPSS_ROOT/config/templates/system_files/hpss.remote.conf.rsyslog.template`

The SSM API provides its own built-in logging mechanism that will separately record access and errors logs into `/var/hpss/log` (default path is `/var/hpss/log/hpss_ssmapi.log` and `/var/hpss/log/hpss_ssmapi_error.log`). Access logs typically capture details of incoming requests, such as timestamps, IP addresses, HTTP methods, response statuses, and processing times. Error logs record critical issues, typically routing errors, exceptions, and failed requests. By default, the SSM API does not manage its own log rotation policy, but the recommended policy can be found at: `$HPSS_ROOT/config/templates/system_files/hpss_ssmapi.logrotated.conf.tmpl`

The Web SSM UI provides a similar logging mechanism to SSM API. Access and error logs will also be located in `/var/hpss/log`, default path is `/var/hpss/log/hpss_webssm.log` and `/var/hpss/log/hpss_webssm_error.log`. A log rotation policy template can be found at `$HPSS_ROOT/config/templates/system_files/hpss_webssm.logrotated.conf.tmpl`.

HPSS rsyslog and logrotate templates can be automatically installed using the

`$HPSS_ROOT/config/configure_logging` tool. This will copy any necessary template files into the proper locations for rsyslog and logrotate.



Note if your site uses syslog-ng, you should not use `configure_logging`. It is important to review the logging and log rotation configuration files before beginning use in a production environment.

20.3.3. Interactions between HPSS and syslog

The default configuration for *rsyslog*, the version of the syslog system that ships with RHEL systems, "escapes" nonprintable control characters in its input by replacing them in its output stream with their octal representation preceded by a hash symbol ("#"). For example, TABs are replaced by "#011" and newlines are replaced by "#012".

If this behavior is not desired, the rsyslog configuration can be changed to convert such characters to spaces instead. To get this effect, update the rsyslog configuration with the following lines:

```
$EscapeControlCharactersOnReceive off
$template HPSS_Format,"%TIMESTAMP% %HOSTNAME %msg:::space-cc%\n"
```

Note that after changing the rsyslog configuration, it will be necessary to "kill -HUP <rsyslog PID>" or restart the rsyslog server.

20.3.4. Configure syslog message size



To avoid message truncation, the maximum message size should be increased from the default.

HPSS log messages, especially IOD/IOR logs, can be extremely large (256 KB). To avoid log message truncation, it is necessary to increase the syslog maximum message size. It is expected that a syslog maximum message size of 256 KB will support nearly all of the log messages HPSS is likely to generate for the foreseeable future without truncation.

`/etc/rsyslog.d/hpss.conf`:

```
...
$MaxMessageSize 256k
...
```

`/etc/syslog-ng/syslog-ng.conf`:

```
options
  {
  ...
    log_msg_size(262144);
  ...
```

```
};
```

20.3.5. Configure multi-part messages

Messages that happen to be larger than the MaxMessageSize are simply truncated in syslog. HPSS can work around this for extremely large messages by dividing the message into multiple syslog calls. The limitations on this are configured with the environment variable HPSS_LOG_MSG_LEN which can be set in `/var/hpss/etc/env.conf`.

HPSS will log any messages larger than HPSS_LOG_MSG_LEN in parts. Each part will have the same header content with the exception of the syslog timestamp. The message payload (`#msg=`) will begin with `MULTIMSG` followed by a unique identifier and the part number. The remainder of the payload contains part of the message that is being logged. The final part is noted by an exclamation point ("!") appended to the `MULTIMSG` tag (`MULTIMSG!`). Below is an example of a three-part message.

```
MULTIMSG 93e0011a34bc234b958e00e3376353b3 0 <message content>
MULTIMSG 93e0011a34bc234b958e00e3376353b3 1 <message content>
MULTIMSG! 93e0011a34bc234b958e00e3376353b3 2 <message content>
```

The default HPSS_LOG_MSG_LEN is 260000. This is slightly less than the default message size in the HPSS syslog template. When setting HPSS_LOG_MSG_LEN, be aware that this does not include the timestamp and hostname prepended by the syslog call. Thus, HPSS_LOG_MSG_LEN should never be set to match the message size limit on your syslog implementation. It should be slightly less to account for this additional space in the messages.

Multi-part messages will appear in the same locations as expected (e.g. SSM), but it currently does not recombine them into the original message. Messages that are not divided into parts will be unaffected. Typically, only IOD and IOR messages are large enough to hit these limits. The hpss_log_formatter(7) tool has been updated to reconstruct messages as it reads the log file. It will print the message once it has all of the parts or when the end of file is reached. Incomplete messages will be marked as such.

20.3.6. Restart Syslog After Configuration Change

Note that after changing the syslog configuration, it will be necessary to restart the syslog service using **service** or **systemctl**. On Linux, using syslog-ng or rsyslog, for example:

```
sudo service rsyslog restart
```

or

```
sudo service syslog-ng restart
```

20.3.7. Managing rate-limiting and duplication

Syslog and Linux in general supports several rate-limiting strategies including dropping messages and collapsing duplicate log messages. These could potentially impact HPSS. Generally for HPSS logs, we would like to eliminate rate-limiting on the Core Server and Mover systems.

The following, when added to `/etc/rsyslog.conf`, will disable rate-limiting for that server.

```
$SystemLogRateLimitInterval 0
$SystemLogRateLimitBurst 0
```

In addition, sometimes the **journal** can rate limit messages. Those settings can be modified in `/etc/systemd/journald.conf`:

```
RateLimitInterval=0
RateLimitBurst=0
```



It is possible to configure the interval and burst to match your system needs; however, this needs to be configured at the high end of the expected volume, or else you may began losing messages. Check your logs during periods of high activity to ensure that messages are not being dropped.

If messages are being dropped, you may begin to see messages such as:

```
imjournal: begin to drop messages due to rate-limiting
Suppressed 1269 messages from /user.slice/user-648.slice
```

If you began to see these messages, adjust your rate limiting in the appropriate configuration files and restart syslog.

In order to disable message reduction via duplication, you can set the following in the HPSS syslog configuration file:

```
$RepeatedMsgReduction off
```

20.3.8. Considerations For SELinux

Systems which are protected by SELinux require additional configuration so to make the HPSS log file accessible to syslog. Otherwise, SELinux will block rsyslog from appending to the HPSS log file.

This can be resolved by giving rsyslog (or your local syslog equivalent) access to the appropriate syslog file targets. For example, if logging to `HPSS.log`, then run:

```
# chcon -t var_log_t HPSS.log
```

20.3.9. Managing log archiving

HPSS takes advantage of the Linux service **logrotate** to facilitate the rotation of its log files. Other Linux services could be used to provide this feature; however, this section will focus on a recommended configuration for **logrotate**, which will provide log rotation and archival services for HPSS.

As logs are archived via **logrotate**, they will be moved into HPSS with a name based upon the HPSS log times that appear in the log. The log file names will appear in one of the following forms:

Table 30. Log file formats

File Name Format	Location	File Status
logfile	File system	File is being filled
logfile.1	File system	File has been rotated once
logfile.1.gz	File system	File has been rotated once and compressed
logfile.YYYY.mmdd.HHMMSS.YYYY.mmdd.HHMMSS	HPSS Log Archive	File has been archived

Note that timestamp date ranges are appended to the log file only when it has been archived into HPSS.

The logrotate policy can set the number of times to rotate log files before deleting them for convenience. When log archiving is included in the logrotate policy, old log files can be retrieved from the archive.

If log archiving is not included in the logrotate policy, then the logrotate policy indicates the number of files to be retained on disk. Once a log file is deleted, it cannot be retrieved.

The administrator needs to consider whether archiving of the log files is needed, what the maximum size for log files should be, how many log files should be retained on disk, and then appropriately set a logrotate policy. See **man logrotate** for details on **logrotate**. Note that if log archiving fails for an HPSS log file, additional log files cannot be archived until that log file is archived.

If **logrotate** is not set up to archive HPSS log files, and then later the administrator decides to enable log archiving, the administrator should first manually archive all HPSS log files beyond the first rotation (for example, `HPSS.log.2` and beyond). Then they may enable log archiving via **logrotate** without losing any existing log information. See hpss_log_archive(7) for usage and options.

If **logrotate** is set up to archive HPSS log files and later log archiving is disabled, then no further HPSS logs will be archived.

The directory to be used for logging in the HPSS archive must be writable by the hpss_log_archive(7) user so that the log archive tool will be able to create log files and new year, month, and day directories as needed. In order for this to occur, both the POSIX-style permissions on the directory must allow write permission and, if account validation is enabled, that user must be in the ACCTVAL table with a default account code. Use the hpss_avaledit(7) utility to check this configuration and change, if necessary.



*If log archiving is set up via **logrotate**, but log file archiving has temporarily stopped working for some reason (for example, due to issues writing to the archive or parts of HPSS being temporarily down), HPSS log files will not rotate and will continue to accumulate until log file archiving begins working again. Care should be taken when temporarily disabling log archiving to allow for log rotation to occur as this could result in the loss of log records. Any logs which are rotated without being archived via **logrotate** can be archived using the hpss_log_archive(7) tool manually.*

20.3.10. Recommendations for logrotate configuration

It is recommended to use the **logrotate** utility for managing the rotation and archiving of HPSS log files. Each HPSS log file should have a logrotate definition in `/etc/logrotate.d/hpss`. See **man logrotate** for information on **logrotate** options. The template file for the recommended logrotate configuration is located in `$HPSS_ROOT/config/templates/system_files/hpss.logrotated.conf.template`.

The template logrotate definition has the following features:

- It will keep five log files locally before deleting them.
- Logs will become eligible for rotation after they are 100 MB in size.
- Logs will be owned by root and have 0600 permissions by default.

There are two script stanzas: prerotate and postrotate. Let's start with postrotate first.

```
    postrotate
    killall -HUP rsyslogd
    /opt/hpss/bin/hpss_log_archive -f $1.1
    endscript
```

Every line after postrotate is a script, which will be run after the log has been rotated (renamed). The first thing we do is SIGHUP rsyslog to stop logging to the rotated file and force it to open the new file.

Now for prerotate:

```
    prerotate
    /opt/hpss/bin/hpss_log_archive -R -f $1.1
    endscript
```

Prerotate handles the situation where a prior postrotate operation failed. The -R (for recovery) parameter tells hpss_log_archive(7) that if the file doesn't exist, just return success. If there is a `logfile.1` out there waiting to be archived though, it will attempt to archive it. If that is successful, then we will log a message indicating we're continuing on with archiving the current log. Note that the call to hpss_log_archive(7) must be the final step in the prerotate and postrotate scripts, or their exit codes retained, so that if the archive step fails, that the active log will continue on *without rotating*.



The hpss_log_archive(7) tool has additional options to control the way that files are written into HPSS.

HPSS rsyslog and logrotate templates can be automatically installed using the `$HPSS_ROOT/config/configure_logging` tool. This will copy any necessary template files into the proper locations for rsyslog and logrotate.



In order to properly handle log archive failures, the call to hpss_log_archive(7) must be the final call in each script, or must be wrapped in a script which retains and returns any errors from hpss_log_archive(7) - the exit code of the final script directive determines logrotate's behavior for prerotate and postrotate, any intermediate errors are ignored.



Note if your site uses syslog-ng, you should not use `configure_logging`. It is important to review the logging and log rotation configuration files before beginning use in a production environment.

A cron job should be configured to run **logrotate**. You may want to have it run every five minutes against the HPSS definition to check the logs, for example. If the log file doesn't match the requirements laid out in the definition, then **logrotate** won't do anything.

```
*/5 * * * * /usr/sbin/logrotate /etc/logrotate.d/hpss
```



The hpss_log_archive(7) tool currently does not work with the `dateext` logrotate option. The `nodateext` option must be used for the hpss_log_archive(7) tool to process rotated log files.

20.3.11. Viewing HPSS logs

HPSS provides the ability to retrieve and examine HPSS log records as a means of analyzing the activity and behavior of HPSS.

To effectively view log records, the user should know what kinds of activities or behaviors are of interest. Since it is possible to configure log policies to include almost all activity in HPSS, in most cases it will be easier to examine and analyze logs by filtering out any records that are not of interest. Since the log files are stored in text format, this can be done using standard UNIX text processing tools.

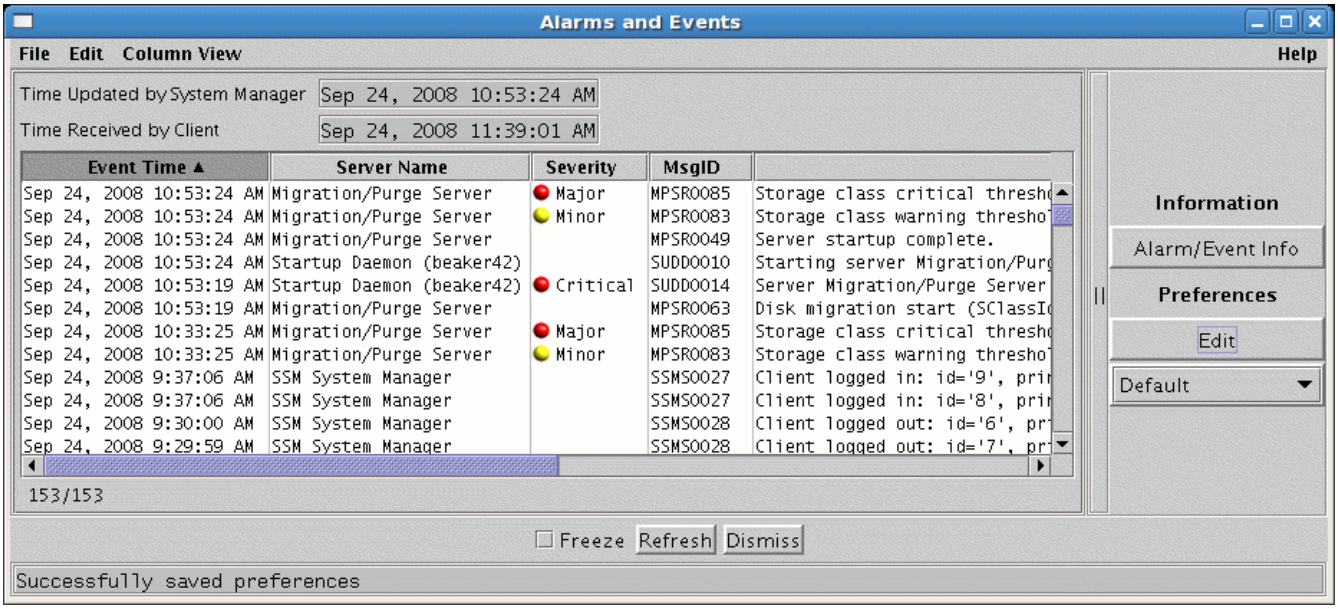
When the HPSS log file, referred to here as `logfile`, reaches the maximum size indicated by its logrotate policy, **logrotate** will rotate all already rotated logs (`logfile.1`-`logfile.N`) such that `logfile.1` becomes `logfile.2` and so on, and anything beyond the logrotate configured rotate count is deleted. Additionally, **logrotate** will rename `logfile` to `logfile.1`. Note that until syslog has been restarted or signaled with SIGHUP, it will keep writing to `logfile.1`. After syslog has switched log files, if log archiving is configured in the logrotate policy, **logrotate** calls hpss_log_archive(7) with the file that was just rotated (`logfile.1`) and hpss_log_archive(7) writes that file into HPSS with its start and end timestamps.

If the log files of interest have rolled off the disk, as long as archiving is configured, they can be retrieved from a date-based directory structure in HPSS. For example, `/log/yyyy/mm/dd`.

20.4. Managing SSM alarms and events

This section describes how to manage SSM alarms and events.

20.4.1. Alarms and Events window

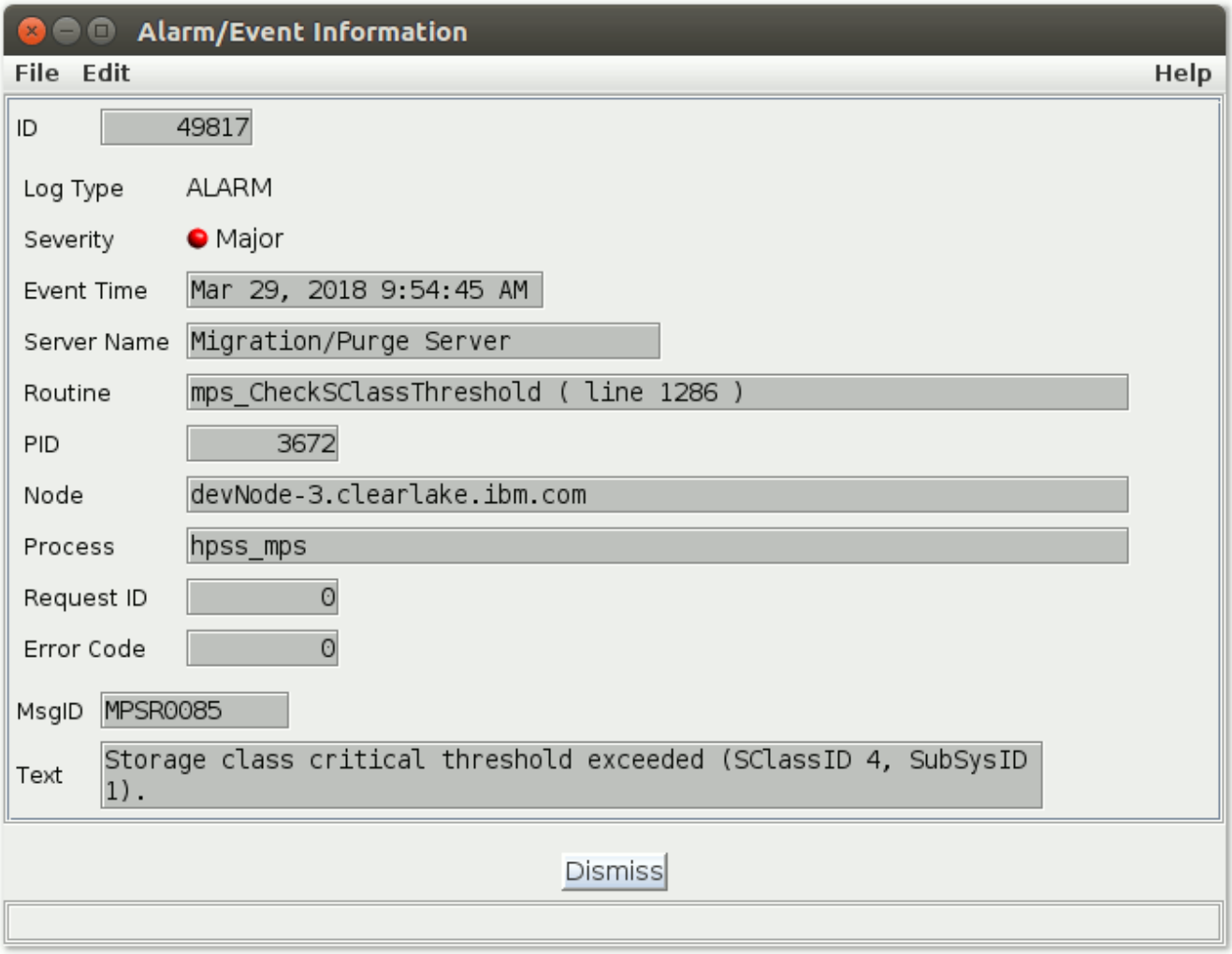


This window displays a number of the most recent alarm and event messages which have been received by SSM. It also allows you to view individual messages in greater detail by selecting the message and clicking the **Alarm/Event Info** button to bring up the *Alarm/Event Information* window.

Field descriptions

This list displays a column for each field shown on the *Alarm/Event Information* window. See Alarm/Event Information for the field descriptions.

20.4.2. Alarm/Event Information



This window displays all the details of the alarm or event selected from the *Alarms and Events* window.

Field descriptions

ID

A sequence number assigned to the log message by SSM. This ID is not used outside of SSM.

Log Type

General class of the message. May be either `ALARM` or `EVENT`.

Severity

For events, this field will be blank. For alarms, it will be one of the following:

- `Warning`
- `Minor`
- `Major`
- `Critical`
- `Indeterminate`
- `Cleared`

These may be accompanied by a color status indicator:

- (Red) for a `Critical` or `Major` alarm
- (Yellow) for a `Minor` or `Warning` alarm
- No indicator for events and other alarm types

Event Time

The date and time the message was generated.

Server Name

Descriptive name of the HPSS server or utility that logged the message.

Routine

Name of the function that was executing when the message was logged.

PID

Process ID of the process that logged the message.

Node

Name of the host where the message was generated.

Process

Name of the process that generated the message.

Request ID

Request ID generated by the process logging the message.

Error Code

The error code associated with the problem underlying the message.

MsgID

The eight-character ID code for this message, consisting of a four-character mnemonic identifying the type of server or subsystem which issued the message followed by a four-digit message number. (The message number is not the same as the **Error Code** field.) Message IDs may be used to look up messages in the *HPSS Error Manual*.

Text

The text of the alarm or event message. Only up to 240 characters are displayed. Look at the `HPSS.log` file to view the entire message.

20.4.3. Diagnosing HPSS problems with alarms and events

Events displayed on the *Alarms and Events* window provide an indication that a significant event has occurred in a server and the event may be of interest to the administrator. An alarm, however, indicates that a server has detected an abnormal condition. The user should investigate the problem as soon as possible to ensure timely resolution. The user may use the information provided by the alarm to obtain further information on the problem as follows:

- Use the alarm message number to look up the alarm information in the *HPSS Error Manual*. For each documented message, the manual provides more detailed information on the problem and its possible source. The manual also provides recommendations on how to resolve the problem.
- If additional information is needed, use the alarm timestamp to identify the log files containing the log messages received around the time the problem was reported. In addition to obtaining the messages logged by the server in question, it may be necessary to obtain log messages from other HPSS servers that interface with the server. Refer to Viewing HPSS logs for more information on locating the HPSS log messages.

20.4.4. Controlling SSM log message handling

This section describes the options available to control SSM handling of HPSS log messages.

This does not include the SSM session log. See the **-S** option on the hpssgui(7) or hpssadm(7) man page for a discussion of SSM session logging.

20.4.4.1. Controlling the System Manager log message cache

By default, the SSM System Manager stores the alarm and event messages it receives in an internal memory cache. This cache can be configured instead to be kept in a disk file by defining the environment variable HPSS_SSM_ALARMS with the desired name of the cache file. The default for HPSS_SSM_ALARMS is defined in `hpss_env_defs.h` as NULL. SSM will revert to the internal memory cache if it cannot access the specified cache file for any reason.

The site may set the HPSS_SSM_ALARMS environment variable to any UNIX file that has read/write access for user *root* on the machine where the SM is to be run (since the SM runs as user *root*). A good path might be `/var/hpss/log/alarm_buffer`, for example.

Using a disk cache makes SSM alarms and events "persistent"; that is, if the System Manager is restarted, SSM will be able to display all old messages still in the disk cache. Caching the alarms and events in memory means the alarm cache starts out empty each time the System Manager is restarted.

By default, 2000 messages are cached before the oldest are discarded. This value can be modified by setting the HPSS_SSM_ALARMS_DISPLAY environment variable to the desired number of messages to be retained and restarting the System Manager.

Once an old message is discarded, it cannot be accessed anymore from the *Alarms and Events* window. Older messages can be accessed by examining the `HPSS.log` file, a log rotated `HPSS.log` file, or by retrieving even older log files from the HPSS archive.

Since the messages are cached and managed by the System Manager, all SSM users connected to the System Manager will see the same messages; however, preferences selected on the *Alarm and Event Preferences* window are applied on a user-by-user basis.

Messages can be delivered to the System Manager and stored in the cache only while the SSM System Manager is running. Any messages generated while the System Manager is down will not be accessible from the HPSS *Alarms and Events* window (but they should be accessible in the local log, central log, or archived log files as long as the logging service processes were running when they

were generated). This means that there can be gaps in the list displayed in the *Alarms and Events* window if the System Manager is shut down and restarted while the rest of HPSS is up.

20.4.4.2. Controlling log messages displayed by hpssgui and hpssadm

The hpssgui(7) and hpssadm(7) programs each keep an internal cached copy of the alarm and event list. The list is maintained by regular polling requests to the System Manager. Review and reference the hpssgui(7) and hpssadm(7) man pages for further details on the settings discussed in this section.

The polling interval is set by the HPSSSSM_UI_ALARM_RATE environment variable, which may be overridden by the **-A** option to the hpssgui(7) or hpssadm(7) startup script. The default is to poll the System Manager every five seconds for new log messages.

Each polling request returns only "new" messages, those which have not already been retrieved by the hpssgui(7) or hpssadm(7) in a previous polling request. If there are no new messages in the System Manager log cache, no messages are returned.

By default, the maximum number of messages retrieved by each polling request is 2000. If there are more than 2000 new messages in the System Manager log message cache, only the 2000 newest will be returned. This means that if new messages are arriving rapidly at the System Manager and the hpssgui(7) or hpssadm(7) polling interval and maximum number of messages values are not set appropriately, there can be gaps in the hpssgui(7) or hpssadm(7) alarm and event list.

The value for the maximum number of messages to return by each polling request may be redefined by the HPSS_SSM_ALARM_GET environment variable or by the **-G** option to the hpssgui(7) or hpssadm(7) startup script. The command line option takes precedence over the environment variable. This option could be used, for example, to reduce the size of each data request on a slow network.

The internal cached alarm and event list is displayed by the hpssadm(7) program by means of its "alarm list" command. This command has a **-c** option to specify how many of the most recent log messages in the internal copy to display. If more messages are requested than exist in the internal list, the full internal list is displayed. See the hpssadm(7) man page for details.

The internal cached alarm and event list is displayed by the hpssgui(7) in the *Alarm and Events* window as described in Alarms and Events window. The maximum number of log messages displayed in the window is set by the HPSS_SSM_ALARM_DISPLAY environment variable, which may be overridden by the **-N** option to the hpssgui(7) startup script. The default is to display at most 2000 messages in the list.

If the maximum number of messages to retrieve (HPSS_SSM_ALARM_GET or **-G** option) is greater than the maximum number of messages to display (HPSS_SSM_ALARM_DISPLAY or **-N** option), it will be reset to the same value as the maximum number of messages to display. This is true for both the hpssgui(7) and the hpssadm(7), although the hpssadm(7) does not use the maximum number of messages to display value for anything else.

If new messages are arriving regularly, the messages displayed by the hpssgui(7) *Alarms and Events* window will constantly be moving downward at set intervals. This can make it difficult to select a specific message if the window is updating frequently. This problem can be minimized by being

careful to click on a message immediately after an update cycle, or it can be eliminated entirely by freezing the display by checking the Freeze check box at the bottom of the *Alarms and Events* window.

If many messages are arriving very rapidly, or if you leave the Freeze button on, it is possible for the display to become stale. A stale message is one which is still visible, but which has been discarded from the cache. If you click on such a message, an error box will appear, telling you that the selected message is no longer available.

Messages are displayed in the order in which they are received by the System Manager. They are not sorted by the event time displayed with each message, so if many messages are generated in a short period of time, it is possible to see messages which are displayed slightly out of chronological order.

Each hpssgui(7) user can filter and sort what is shown on his own *Alarms and Events* window independently of any other hpssgui(7) user. See SSM list preferences for information on filtering and sorting alarms. Filtering and sorting of the *Alarms and Events* list are not yet available for hpssadm(7) users.

20.4.5. Media Access Logging

HPSS logs individual media access for I/O operations. This feature is always on, but the logging only occurs if the INFO log is enabled for the Core Server.

This feature allows an administrator to understand which files may have been written to or read from a particular disk or tape volume, which tape drives may have been used to read or write a file, and when those operations occurred. This can assist in certain data investigations, such as discovering files which could have been impacted by bad firmware.

Fields in the start access log:

Media Access

Media access logs contain an initial tag to differentiate them for filtering or redirection.

Operation

The operation that is beginning; for example, read, write, stage, copy, or migration.

BFID

The Bitfile ID for the file being logged.

Fileset

The fileset name for the file being logged.

Path

The path to the file from the fileset root.

Fields in the end access log:

Media Access

Media access logs contain an initial tag to differentiate them for filtering or redirection.

Operation

The operation that is completing; for example, read, write, stage, copy, or migration.

Media Read

In the format <Type><DeviceId>:<Volume>. Indicates the type (`T` for tape, `D` for disk) of media, the device ID that was used, and the one or more volumes read. If multiple volumes were read they will be separated by a comma.

Media Write

In the format <Type><DeviceId>:<Volume>. Indicates the type (`T` for tape, `D` for disk) of media, the device ID that was used, and the one or more volumes written. If multiple volumes were written they will be separated by a comma.

BFID

The Bitfile ID for the file being logged.

Fileset

The fileset name for the file being logged.

Path

The path to the file from the fileset root.

Result

Indicates the result as a string along with the numeric error code.

Media Access logs look like:

```
May 30 10:02:07.793192 hpss_core(HPSS)[756]:: #msgtype=INFO
#server=Core Server@server #func=bfs_StageThread
(line 1478) #rc=0 #msgid=CORE2188 #reqid=6068557350692913156
#msg=Media Access : Stage begin : BFID
x'00000539070000000801E841C3C0A743B01E34' : FilesetRoot.29 :
./path/to/file
May 30 10:02:07.793801 hpss_core(HPSS)[756]:: #msgtype=INFO
#server=Core Server@server #func=bfs_StageCleanup
(line 3046) #rc=0 #msgid=CORE2189 #reqid=6068557350692913156
#msg=Media Access : Stage end : read {T4:JD021700} : wrote
{D2:MY000200,D1:MY000100} : BFID
x'00000539070000000801E841C3C0A743B01E34' : FilesetRoot.29 :
./path/to/file : Result No error (0)
```

Media access logs exist for file read, write, migrate, batch migrate, stage, async stage, and copy operations. Media access logging is currently unavailable for Repack.

20.5. Log files of various types - a comprehensive list

HPSS generates log files of its own; however, a number of log files of various types are generated in the operation of an HPSS system. There are also files that are not log files, strictly speaking, but which grow over time and have the potential to fill the disk where they reside. This section attempts to provide a comprehensive list of such files along with guidance for managing them.

/var/hpss/log/HPSS.log:

Default location for HPSS system logging. Log rotation and archiving should be configured with this as described in Recommendations for logrotate configuration.

/var/hpss/log/hpss_ssmapi.log:

Access logging for SSM API. Captures details of incoming requests, such as timestamps, IP addresses, HTTP methods, response statuses, and processing times. Log rotation and archiving should be configured with this as described in Recommendations for logrotate configuration.

/var/hpss/log/hpss_ssmapi_error.log:

Error logging for SSM API. Logs critical issues, typically routing errors, exceptions, and failed requests. Log rotation and archiving should be configured with this as described in Recommendations for logrotate configuration.

/var/hpss/log/hpss_webssm.log:

Access logging for Web SSM UI. Captures details of incoming requests, such as timestamps, IP addresses, HTTP methods, response statuses, and processing times. Log rotation and archiving should be configured with this as described in Recommendations for logrotate configuration.

/var/hpss/log/hpss_webssm_error.log:

Error logging for Web SSM UI. Logs critical issues, typically routing errors, exceptions, and failed requests. Log rotation and archiving should be configured with this as described in Recommendations for logrotate configuration.

/var/hpss/adm/hpssd.failed_server:

A record is written to this file each time an HPSS server exits abnormally. Normally, the growth of this file will be very slow, but eventually it could fill the disk, so it should be

monitored and removed if it threatens to become too large.

/var/hpss/adm/core:

If an HPSS server suffers a severe internal error and crashes, a diagnostic core dump will be placed in `/var/hpss/adm/core/<Server-Name>/core.<pid>`. These core files can be very large and if enough of them accumulate, the disk could be filled. This area should be checked periodically and cleaned out if necessary.

/var/hpss/ftp/adm/hpss_ftpd.log:

PFTP uses `/var/hpss/ftp/adm/hpss_ftpd.log` for Client API debugging. Debugging is enabled by setting

```
Client API Verbose Value = <n>
```

in `HPSS.conf`, where <n> is the debugging level desired. The file input is provided directly by the Client API. If debugging is left on, this file can also grow without bounds, so it should only be used for debugging and removed when debugging is completed.

/var/hpss/hpssdb/sqllib/db2dump/db2diag.log:

DB2 writes diagnostic messages to `db2diag.log`. If this file gets too large, it can be renamed and optionally compressed for archival. Once a copy has been archived, the renamed file should be removed from the system.

To manage the size of this file, you can periodically archive and remove it or use a log rotation tool of your choice (for example, `/usr/sbin/logrotate`).

/var/hpss/log/krb5kdc.log:

Kerberos information and errors are written to this file. Its location is defined in the Kerberos configuration, usually in `/etc/krb5.conf`. This file will only be generated if you use Kerberos. To manage its size, you can periodically archive and remove it or use a log rotation tool of your choice (for example, `/usr/sbin/logrotate`).

/var/hpss/mps/<report-files>:

The Migration/Purge Server will write daily reports in `/var/hpss/mps` if the MPS Report File Name field in the Migration/Purge Server Specific Configuration is not empty. Unless this directory is monitored and old reports are removed, these report files have the potential of

filling the disk.

/var/hpss/tmp/gasapi.diag:

See the description of `secapi.diag` below. This file behaves exactly the same way.

/var/hpss/tmp/secapi.diag:

Normally, on HPSS systems, this file does not exist. It can be used to troubleshoot security issues in HPSS if needed. To use it, issue the command

```
touch /var/hpss/tmp/secapi.diag
```

Once the file exists, security logs will be appended to it as they arise. The file contents can be viewed with any text file viewer.

A command like the following will truncate the file:

```
/bin/echo -n "" > /var/hpss/tmp/secapi.diag
```

If this file exists and HPSS is running, the file should not be deleted. Deleting a file while one or more processes have it open will remove the file's directory entry but leave the file's data segments on disk still allocated and occupied. As the running processes continue writing to it, the file will grow invisibly, eventually consuming all available disk space.

If this should happen, to recover, it is necessary to shut down all HPSS processes. Once all the processes holding the phantom file open terminate, the operating system will notice that the disk blocks are associated with a file that no longer exists and recover them.

/var/log/wtmp, /var/adm/wtmp:

UNIX writes a record to this file any time a user logs in to or out of the system. These records can be viewed with the **last(1)** utility. If this file grows too large and needs to be removed, with the appropriate privilege, it can be truncated by redirecting the output of a silent command (that is, one that produces no output):

```
cat /dev/null > /var/log/wtmp
```

To manage the size of this file, you can periodically archive and remove it or use a log rotation tool of your choice (for example, `/usr/sbin/logrotate`).

/var/hpss/tmp/scsi.diag:

device_scan(7), the SCSI PVR, the Mover, and other tools that interact with the HPSS SCSI layer may log SCSI diagnostic messages to this log file. To enable this logging, touch the `scsi.diag` file. This logging can be helpful in troubleshooting issues in the SCSI or device layer. This file will grow without bound. In order to cause all HPSS processes to stop logging to this file, touch `/var/hpss/tmp/noscsidiag`. HPSS processes logging to `scsi.diag` will stop logging within 30 seconds and close the file. This allows the `scsi.diag` file to be cleaned up or rotated easily. Removing the `/var/hpss/tmp/noscsidiag` file resumes all logging for any HPSS process that is logging SCSI diagnostics.

/var/hpss/tmp/<PID>.diag:

Several HPSS servers — including the Core Server, PVL, PVR, SSM, Mover, RAIT Engine, SD, and GK — will dump out internal runtime information to a log file named after the server instance's Process ID (PID). The server state dump information typically includes generic information such as RPC diagnostics and cached SQL queries. After those sections, the servers output custom listings of internal runtime information.

Sending a `SIGHUP` signal to one of the listed HPSS server processes will cause the state dump file to be written to `${HPSS_PATH_TMP}`. These log files are typically most useful during investigation and debugging. It can be particularly helpful to generate one of these state dump files while a problem is occurring and then provide the file to HPSS Support.

API debug log:

The Client API can generate debug logging to assist with application and configuration issues. For a full discussion of Client API logging, consult the *Client API Tutorials and Best Practices* chapter of the *HPSS Programmer's Reference*.

There are three ways to enable the API debug log:

- Add HPSS_API_DEBUG=7 and HPSS_API_DEBUG_PATH=/path/to/log to `$HPSS_PATH_ETC/env.conf`.
- Export HPSS_API_DEBUG=7 and HPSS_API_DEBUG_PATH/path/to/log before running the program.
- Update the `hpss.api.resource` file in $HPSS_PATH_TMP. The format is:
  7 /path/to/log

The logging begins for environment variable methods the next time a Client API instance is initialized. In the resource file case, the logging will not begin immediately, but the next time the API reads the resource file.

The logging stops for environment variable methods when the API instance ends. For resource logging, the resource file can be removed or the log level changed to "0" to disable

logging. Logging will be disabled the next time the API reads the resource file. The resource file impacts *all* running instances on the system so care should be used before using this method. The log location can also be specified as `stdout`.

The API resource file is checked every 30 seconds.

Authentication files:

Some HPSS utilities backup certain authentication files before changing them. These files reside in `/var/hpss/etc` and are named `passwd`, `group`, and `shadow`. The backups have names of the form <filename>.<epoch> where <epoch> is a time stamp consisting of the number of seconds since the beginning of 1970. If these files see a lot of activity, it may be necessary to periodically remove the backup files to keep the disk from filling.

Mover Debug Logging:

To control which Movers generate debug output, what message types are printed, the format of the output, and where the debug output messages will be written you must create a `MoverDebug.conf` file in the $HPSS_PATH_TMP directory (the default is `/var/hpss/tmp/MoverDebug.conf`). This file should be created on the server on which the hpss_mvr_adm process is running. After this file is created, any Movers added to `MoverDebug.conf` will need to be restarted.

The format of the file is:

'MOVER DESCRIPTIVE NAME' MASK FILE_PATH

where:

- MOVER DESCRIPTIVE NAME is the descriptive name of the Mover for which to enable debugging. It *must* be enclosed in single quotes (').
- MASK is a hexidecimal string specifying the format and types of messages to output. The string is obtained by ORing the desired flags (found in `mvr_shmem.h`). Using 0x00FFFFFF will enable all Mover logging options.
- FILE_PATH is either the pathname of a file that will contain the output or the string *stdout*, in which case the output will be written to stdout of the tty associated with the root shell that started the Mover or the SSM.

Any line starting with a # symbol is ignored.

Example:

```
'DiskMover1 (gauss)' 0x50027  /var/hpss/tmp/DM1.out
'TapeMover  (T10K)' 0x0F003F  stdout
```

The Mover debug configuration file is read periodically (a few times a minute). Note that *only*

the MASK value can be changed without restarting the Mover.

20.6. Disk Mover & VV I/O Metrics

Metrics about disk mover and VV I/O operations are available in JSON formatted log files. These files are regularly trimmed in order to ensure that they don't fill the file system where they are created. See the section about HPSS Disk Mover and VV Load-Balancing for details.

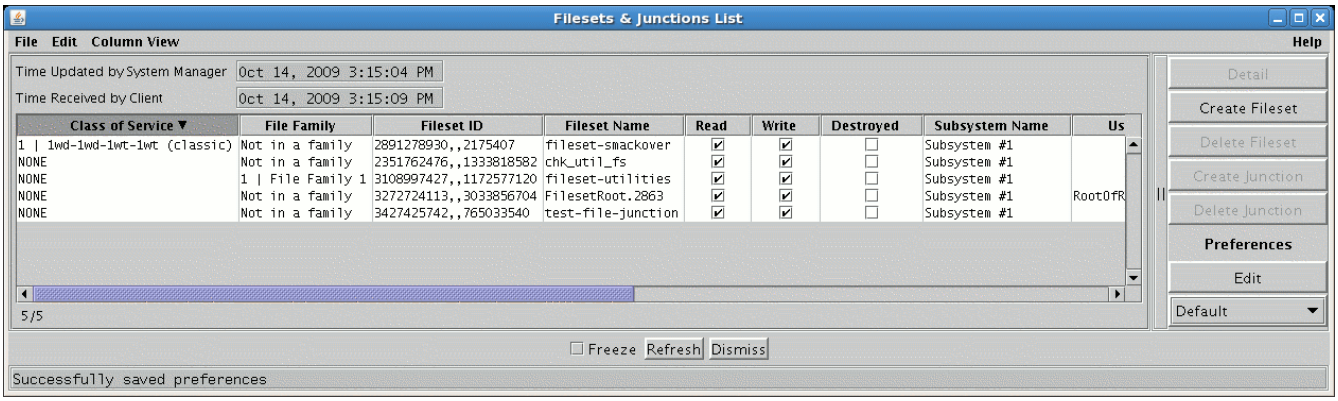
Chapter 21. Filesets and junctions

A fileset is a logical collection of files that can be managed as a single administrative unit, or more simply, a disjoint directory tree. A fileset has two identifiers: a human readable name, and a numeric fileset ID. Both identifiers are unique to a given HPSS realm. Filesets are often used for HPSS name space administration. For example they can be used if a subtree of the HPSS name space needs to be assigned to a particular file family or Class of Service.

A junction is a Core Server object, much like a symbolic link, that is used to point to a fileset. This fileset may belong to the same Core Server or to a different Core Server. When pointing to a different Core Server, junctions allow HPSS users to traverse to different subsystems.

When a Core Server initializes for the first time on an empty database, it creates the *root node* fileset. This is the fileset that contains the directory named "/" (forward slash). For many sites this may be the only fileset needed and no other filesets will ever be created.

21.1. Filesets & Junctions List



This window displays information about the filesets and junctions that are configured in the HPSS system. From this window, detailed information about existing filesets and junctions can be viewed, new filesets and junctions can be created, and existing filesets and junctions can be deleted.

To open the window, from the *HPSS Health and Status* window **Monitor** menu select **Filesets & Junctions**.

There may be multiple entries in the list for a given fileset meaning that there are multiple junctions that point to the same fileset. Each entry for a given fileset will have a unique Junction Name. If the Junction Name is blank, then there are no junctions that point to the particular fileset.

The lsfilesets(7), lsjunctions(7) and lshpss(7) utilities may also be used to list all the filesets and junctions in the system. For more information, refer to the man pages for each utility.

Field descriptions

Filesets & Junctions List

Class of Service

The name of the Class of Service to which the fileset is assigned. If this field contains "NONE", the fileset has not been assigned to a Class of Service.

File Family

The name of the file family to which the fileset is assigned. If this field contains `Not in a family`, the fileset has not been assigned to a family.

Fileset ID

The ID number which identifies the fileset. A fileset ID is displayed as two double-comma-separated unsigned integer numbers.

Fileset Name

The unique name which has been assigned to the fileset.

Read

If checked, the fileset is available for reading.

Write

If checked, the fileset is available for writing.

Destroyed

If checked, the fileset cannot be written to or updated.

Subsystem Name

The subsystem name to which the fileset has been assigned. This is the subsystem of the Core Server which controls the fileset.

User Data

This field is available for storing up to 128 bytes of data. The information can be ASCII or binary.

Directories

The number of directories in this fileset.

Files

The number of files in this fileset.

Hard Links

The number of hard links in this fileset.

Junctions

The number of junctions in this fileset.

Sym Links

The number of symbolic links in this fileset.

Junction Name

The name of a junction that points to this fileset. This field may be blank if there are no junctions configured to point to this fileset.

Parent Fileset

The name of the fileset in which the junction resides. This field will be blank if the Junction Name is blank.

Buttons

Detail

Display the *Core Server Fileset Information* window for the selected filesets. This button is only enabled if one or more items are selected in the list.

Create Fileset

Display the *Create Fileset* window so that a new fileset may be created. This button is enabled all the time.

Delete Fileset

Delete the selected filesets. This button is only enabled if one or more items are selected in the list.

Create Junction

Create a junction to the selected fileset. This button is only enabled when there is exactly one item selected in the list. The fileset name and ID from the selected fileset will be used to fill in the information on the *Create Junction* window. The user will only have to supply the Junction Name on the *Create Junction* window to create the new junction.

Delete Junction

Delete the selected junctions. This button is only enabled if one or more items are selected in the list.

21.2. Creating an HPSS fileset

This section provides information on how to create HPSS filesets. Only the HPSS root user and SSM principal are allowed to create filesets.



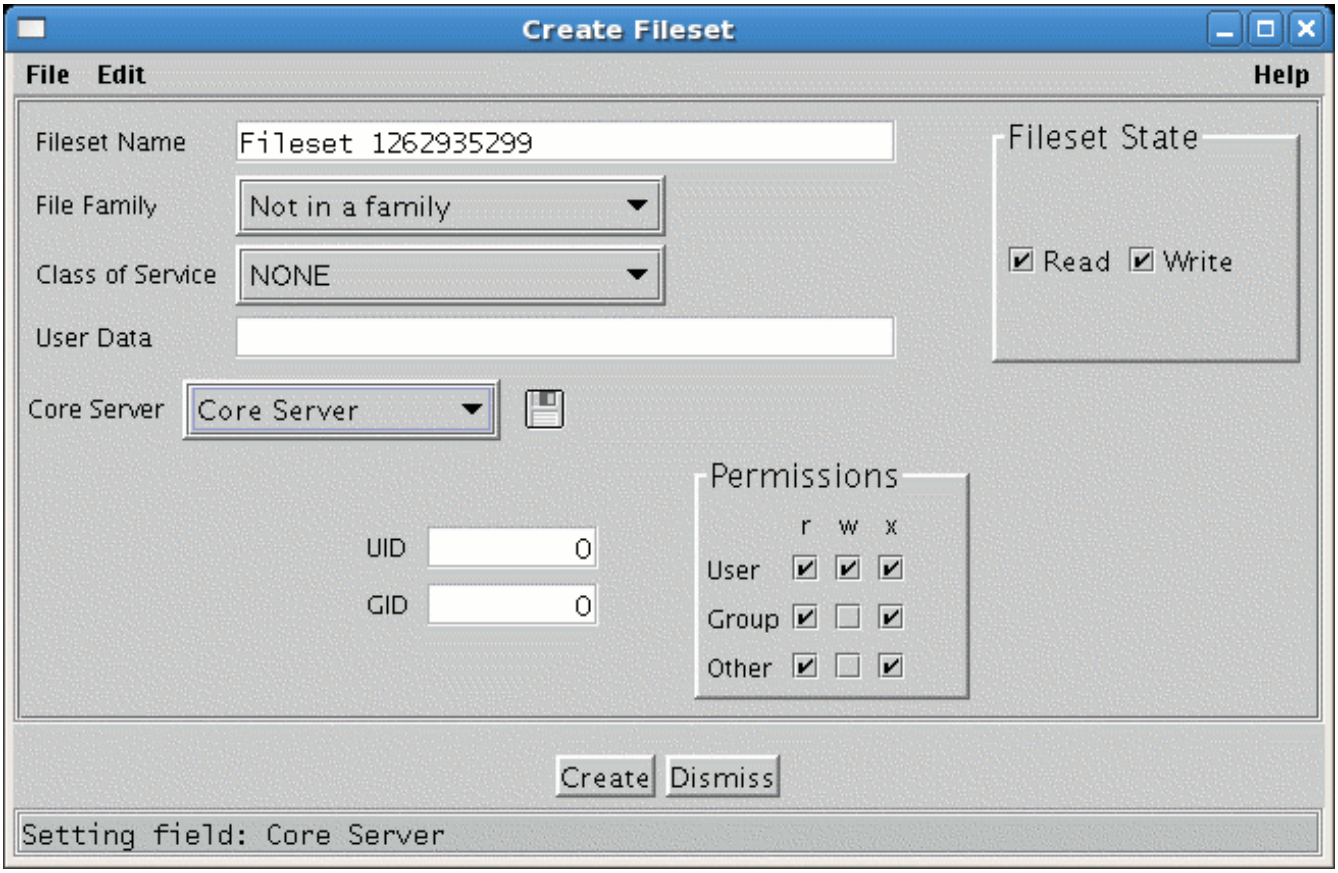
In order to successfully perform fileset administration, the DB2 Helper Program must be bound. See the Generate and bind the DB2 helper program section for more information.

_An HPSS fileset can be created by using the Create Fileset SSM window, or by using the crtfileset(7) utility.

SSM automatically generates a fileset ID for the new fileset.

After a fileset has successfully been created, a junction will need to be created to get to the new fileset name space from the root node fileset name space.

21.2.1. Create Fileset window



The *Create Fileset* window is used to create new HPSS filesets.

To open the window, from the *HPSS Health and Status* window **Monitor** menu select **Filesets & Junctions** to display the *Filesets & Junctions List*, then click the **Create Fileset** button.

After creating a new HPSS fileset, a new junction will be needed to connect this new fileset to the HPSS name space. See Creating a junction for more information.

Field descriptions

Fileset Name

The name to be assigned to the fileset. This name must be unique to the realm in which HPSS resides.

Fileset State

The state of the fileset. If **Read** is ON, the fileset will be available for reading. If **Write** is ON, the fileset will be available for writing.

File Family

The name of the file family assigned to this fileset. If the file family is to be other than the default, the file family must have been previously created.

Class of Service

The name of the Class of Service assigned to this fileset. This menu will display the available Classes of Service.



Advice

The **File Family** and **Class of Service** fields are optional. They provide the capability to force all data stored in the fileset to be assigned to a particular file family or Class of Service or both.

User Data

Any name or other data (up to 128 bytes) that the administrator wishes to associate with the fileset. The information can be ASCII or binary, although ASCII is easier to work with in this field. HPSS does not use this field in any way; it is strictly for user convenience in annotating the fileset. The field may be left blank.

The field is displayed as printable ASCII characters where possible. Nonprintable bytes are displayed in backslash-octal notation, where each byte is shown as a backslash ("\") followed by a three-digit octal number. For example, a tab character (decimal value 9) would be displayed as "\011". Backslash characters are themselves displayed as two backslashes ("\\"). Trailing null (that is, zero) bytes are not displayed.

To modify this field, enter data in the same format. Printable characters (except for the backslash) can be entered normally. Backslash characters must be entered as "\\". Nonprintable characters must be entered in backslash-octal notation. You need not specify leading zeros on the octal numbers, EXCEPT when the nonprintable byte is followed by a printable octal digit character (0 - 7). In that case you must specify all three digits of the octal number to distinguish it from the printable character following.

Core Server

The name of the Core Server which will create and manage the fileset.

UID

The user ID identifying the user owning the root node of the fileset.

GID

The group ID identifying the principal group owning the root node of the fileset.

Permissions

The initial UNIX-style permissions to be assigned to the root node of the fileset. There are nine check boxes arranged in a matrix with the columns specifying "r" (read), "w" (write) and "x" (execute) permissions, and the rows specifying the three classes of users to which the permissions apply (User, Group, and Other). If a check box contains a check, it means that access is permitted. For example, if the check box in the "x" column and the Group row contains a check, it means that Group users have execute access.

Buttons

Create

Creates the fileset. If the create succeeds, a success message will appear on the status line at the bottom of the window; otherwise an error message will appear.

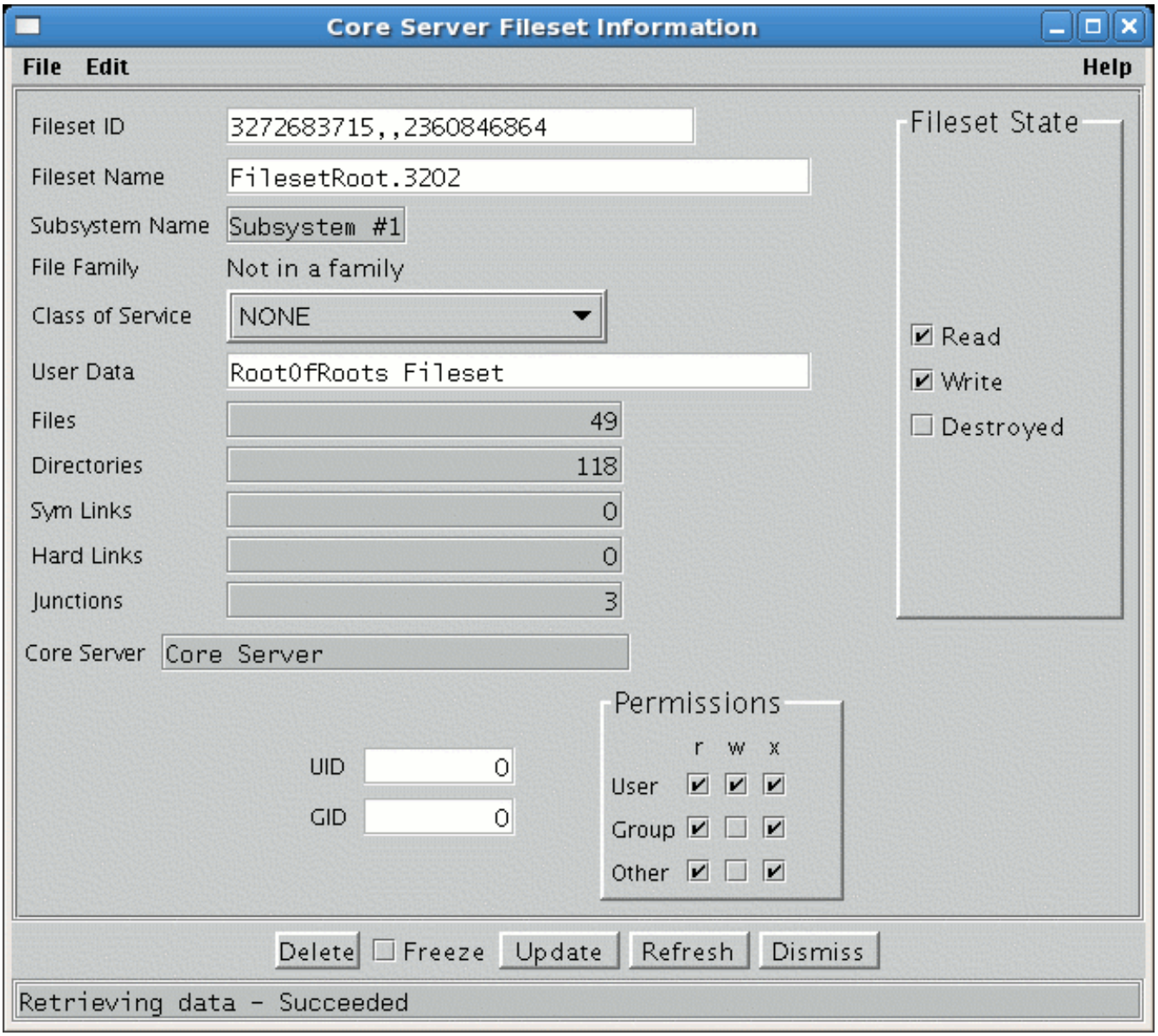
Dismiss

Closes the window.

21.3. Managing existing filesets

This section describes how to look up information on, modify, or delete filesets.

21.3.1. Core Server Fileset Information window



This window allows an administrator to view, update, and delete the Core Server information associated with a fileset. This information is acquired from the Core Server. While this window remains open, it will automatically update whenever change notifications about any field (except UID, GID, and permissions) are received from the Core Server.

To open the window, from the *HPSS Health and Status* window **Monitor** menu select **Filesets & Junctions** to display the *Filesets & Junctions List*, select one or more filesets, then click the **Detail** button.

Any changes made to fields on this window are sent directly to the Core Server when the

administrator clicks on the **Update** button; the changes are effective upon successful update. To mitigate automatic updates overwriting changes in progress, it would be wise to freeze this window by checking the **Freeze** check box before making changes.

The **Fileset State** can be used to prevent any changes to fileset data or to fileset metadata. Changing the state to just Read (unchecking the **Write** check box) will allow reading, but no changes will be allowed to the data or metadata. Changing the state to Destroyed will prevent both reading and writing.

Field descriptions

Fileset ID

The ID number which identifies the fileset. A fileset ID is displayed as two double-comma-separated unsigned integer numbers. A new Fileset ID can be entered as two double-comma-separated unsigned integer numbers, as two single-period-separated unsigned integer numbers, as a 64-bit hexadecimal number that begins with *0x*, or as an unsigned 64-bit number.

Fileset Name

The unique name which has been assigned to the fileset. The administrator can change the **Fileset Name** as long as the new name is unique to the realm in which HPSS resides.

Subsystem Name

The subsystem name to which the fileset has been assigned. This is the subsystem of the Core Server which controls the fileset.

File Family

The name of the file family to which the fileset is assigned. If this field contains `Not in a family`, the fileset has not been assigned to a family.

Class of Service

The name of the Class of Service to which the fileset is assigned. If this field contains `NONE`, the fileset has not been assigned to a Class of Service.

Fileset State

This set of check boxes displays fileset states, and allows them to be changed.

Read

If checked, the fileset is available for reading.

Write

If checked, the fileset is available for writing.

Destroyed

If checked, the fileset cannot be written to or updated; in fact, it is not possible to set any of the attributes except the **Fileset State** attribute.

User Data

This field is available for storing up to 128 bytes of data. The information can be ASCII or binary,

although ASCII is easier to work with in this window.

The field is displayed as printable ASCII characters where possible. Nonprintable bytes are displayed in backslash-octal notation, where each byte is shown as a backslash ("\") followed by a three-digit octal number. For example, a tab character (decimal value 9) would be displayed as "\011". Backslash characters are themselves displayed as two backslashes ("\\"). Trailing null (that is, zero) bytes are not displayed.

To modify this field, enter data in the same format. Printable characters (except for the backslash) can be entered normally. Backslash characters must be entered as "\\". Nonprintable characters must be entered in backslash-octal notation. It is not necessary to specify leading zeros on the octal numbers, EXCEPT when the nonprintable byte is followed by a printable octal digit character (0 - 7). In that case, all three digits of the octal number must be specified.

Files

The number of files in this fileset.

Directories

The number of directories in this fileset.

Sym Links

The number of symbolic links in this fileset.

Hard Links

The number of hard links in this fileset.

Junctions

The number of junctions in this fileset.

Core Server

The name of the Core Server that handles this fileset.

UID

The user ID identifying the user owning the root node of the fileset.

GID

The group ID identifying the principal group owning the root node of the fileset.

Permissions

The UNIX-style permissions assigned to the root node of the fileset. There are nine check boxes arranged in a matrix with the columns specifying "r" (read), "w" (write) and "x" (execute) permissions, and the rows specifying the three classes of users to which the permissions apply (User, Group, and Other). If a check box contains a check, it means that access is permitted. For example, if the check box in the "x" column and the Group row contains a check, it means that Group users have execute access.

Buttons

Delete

Requests deletion of the currently displayed fileset. A confirmation will be requested before the request is sent to the Core Server. If the deletion is successful, a message will appear on the status line at the bottom of the window and all fields and buttons will be disabled (except for the **Dismiss** button). Note that only empty filesets may be deleted.

Freeze

Clicking on this check box freezes the window; in other words, all automatic updates to the window are suspended. Toggling the check box again returns the window, and the check box, to their normal conditions. Any accumulated changes that occurred while the window was frozen are immediately displayed.

Update

Requests that the currently displayed fileset information replace the fileset information currently kept by the Core Server.

Refresh

Requests that the current information about this fileset be fetched from the Core Server.

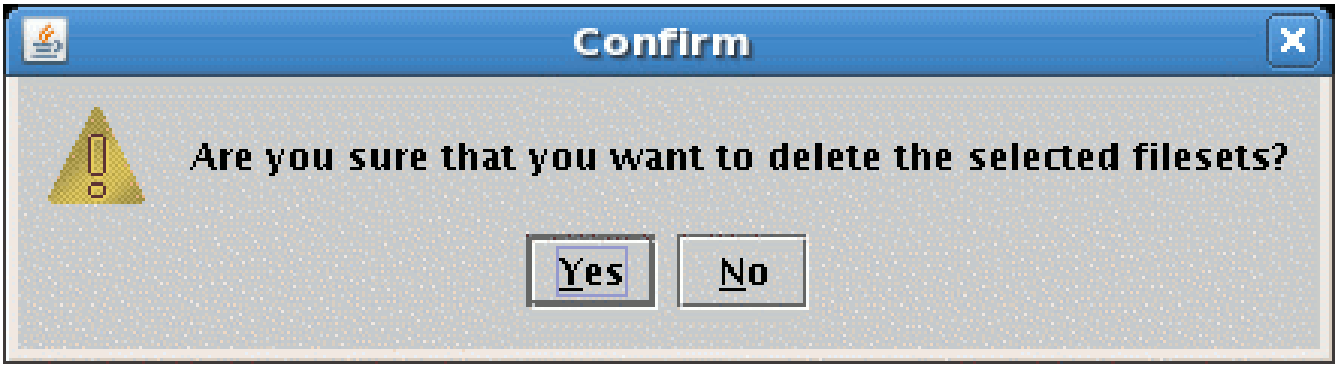
Dismiss

Closes the window.

21.4. Deleting filesets

Filesets must be empty in order to be deleted. The Core Server will reject the deletion request if the fileset is not empty.

Filesets may be deleted by selecting the filesets on the *Filesets & Junctions List* and clicking the **Delete** button. The following confirmation will be displayed:



To continue with the fileset deletion, click **Yes**.

Filesets may also be deleted from the *Core Server Fileset Information* window using the **Delete** button.

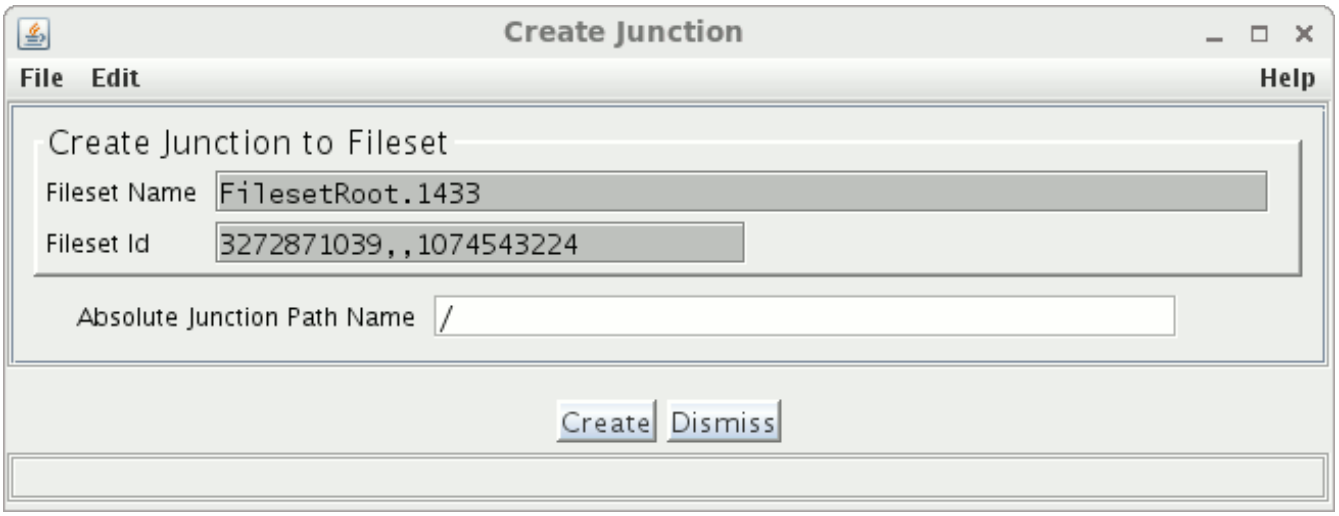
21.5. Creating a junction

Only the HPSS root user and SSM principal are allowed to create junctions.

A junction is a name space object that points to a fileset and is similar to a persistent UNIX mount point. The fileset pointed to may reside in another subsystem.

Junctions can be created using SSM or by using the utility routine crtjunction(7).

21.5.1. Create Junction window



This window allows you to create a junction named by the **Absolute Junction Path Name** to the specified fileset. HPSS defines a junction as a name space object that *points* to a directory. A fileset is an independent directory subtree, managed as a unit. The fileset to which the junction will point may be managed by the same Core Server where the junction is being created or it may be managed by another HPSS Core Server. HPSS junctions are similar to persistent UNIX mount points.

Although HPSS allows junctions to any directory, this window can only be used to create junctions to directories that are also filesets.

Field descriptions

Fileset ID

The ID number which identifies the fileset to which the junction will point. This field is filled in from the Fileset ID selected on the *Filesets & Junctions List*.

Fileset Name

The unique name which identifies the fileset to which the junction will point. This field is filled in from the Fileset Name selected on the *Filesets & Junctions List*.

Absolute Junction Path Name

The absolute path name, in the HPSS name space, of the junction that is to be created. If the path name to the junction is not valid, the creation will fail.

Buttons

Create

Creates the junction. If the create succeeds, a success message will appear on the status line at the bottom of the window; otherwise an error message will be displayed. At this point, data can

be entered for another junction, or the window can be dismissed.

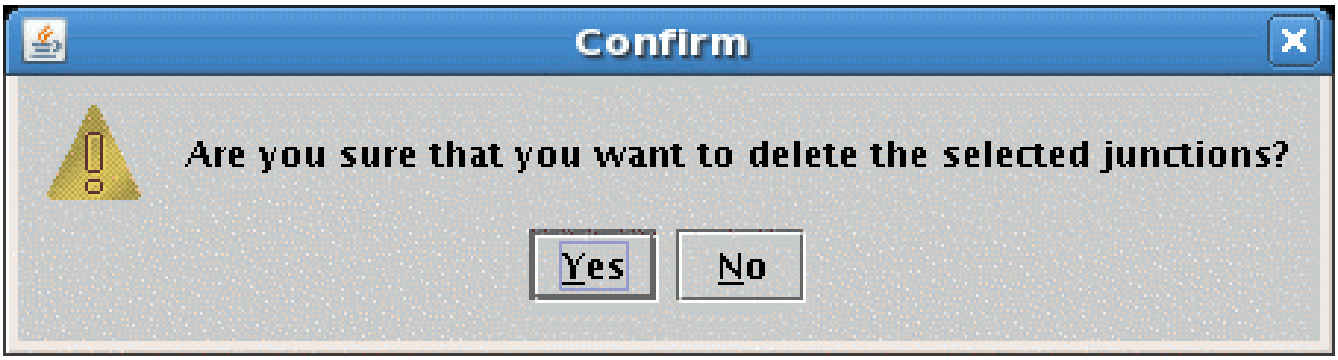
Dismiss

Closes the window.

21.6. Deleting a junction

Junctions can be deleted using SSM or by using the utility routine deljunction(7).

To delete a junction using SSM, select the junctions to be deleted from the *Filesets & Junctions List* and click the **Delete Junction** button. You will be asked to confirm the deletion with the following dialog:



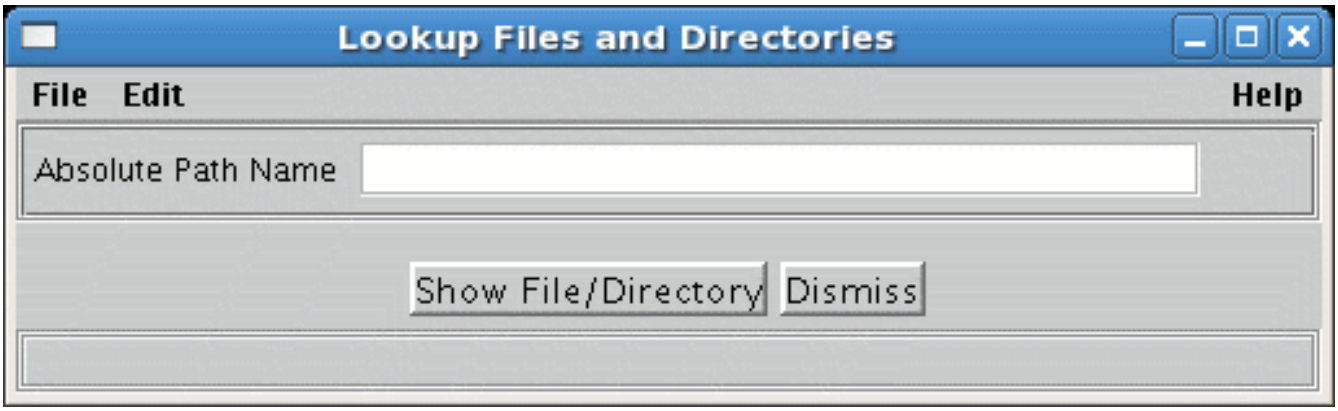
To continue with the junction deletion, click **Yes**.

Chapter 22. Files, directories and objects by SOID

This chapter describes two ways to display basic information about files and directories stored in HPSS. Starting with a fully qualified path name, you can look up a file or directory in the system and display information about it. In other cases, your starting point may be a SOID, a Storage Object ID, which is the internal computer generated name for files, directories, and virtual volumes. Some utility programs report the names of objects of interest as SOIDs.

SOIDs are more difficult to deal with manually as they are not intended to be used that way. But they contain all the necessary information to identify the server, object, and object type, so they can be used to zero-in on a very specific internal data structure, such as a bitfile or virtual volume, that would otherwise be very difficult to isolate and display.

22.1. Files and Directories window



This window is reached through **Monitor** > **Lookup HPSS Objects** > **Files & Directories**. The pathname of either an HPSS file or directory may be entered. Use the **Show File/Directory** button to display specific information.

Field descriptions

Absolute Path Name

The absolute path name (or fully qualified path name), in the HPSS name space, of the file or directory that is to be displayed. If the path name to the file or directory is not valid, then lookup will fail. If the path name is to a symbolic link or junction, then the object displayed will be the object pointed to by the symbolic link or junction.

Buttons

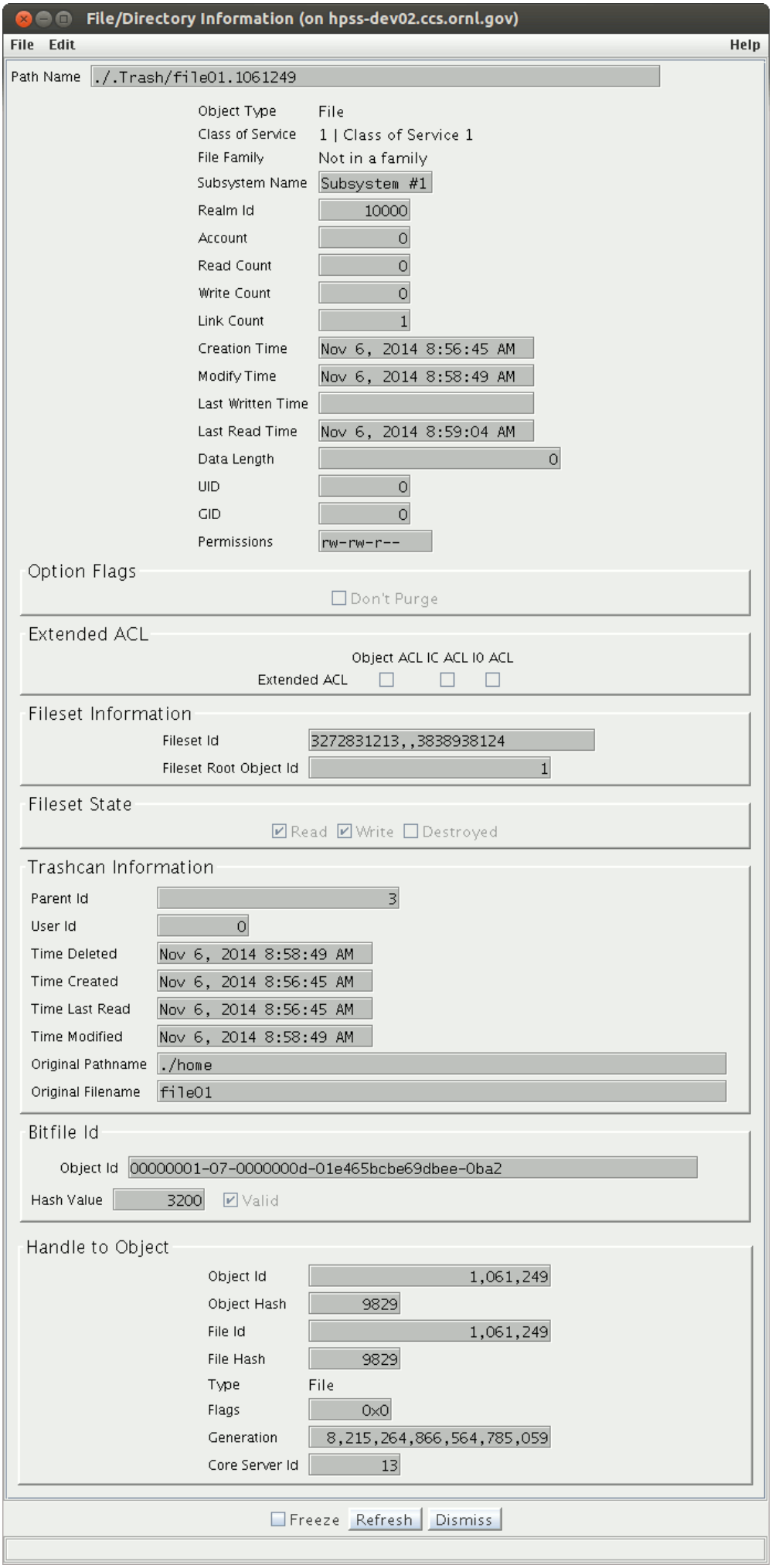
Show File/Directory

If the Absolute Path Name exists, clicking on this button will open the *File/Directory Information* window. The *Lookup Files and Directories* window will remain open so that the administrator may specify another file or directory.

Dismiss

Closes the window.

22.1.1. File/Directory Information window


File/Directory Information (on hpss-dev02.ccs.ornl.gov)
File Edit Help
Path Name ./.Trash/file01.1061249
Object Type File
Class of Service 1 | Class of Service 1
File Family Not in a family
Subsystem Name Subsystem #1
Realm Id 10000
Account 0
Read Count 0
Write Count 0
Link Count 1
Creation Time Nov 6, 2014 8:56:45 AM
Modify Time Nov 6, 2014 8:58:49 AM
Last Written Time
Last Read Time Nov 6, 2014 8:59:04 AM
Data Length 0
UID 0
GID 0
Permissions rw-rw-r--
Option Flags
Don't Purge
Extended ACL
Object ACL IC ACL IO ACL
Extended ACL
Fileset Information
Fileset Id 3272831213,,3838938124
Fileset Root Object Id 1
Fileset State
Read Write Destroyed
Trashcan Information
Parent Id 3
User Id 0
Time Deleted Nov 6, 2014 8:58:49 AM
Time Created Nov 6, 2014 8:56:45 AM
Time Last Read Nov 6, 2014 8:56:45 AM
Time Modified Nov 6, 2014 8:58:49 AM
Original Pathname ./home
Original Filename file01
Bitfile Id
Object Id 00000001-07-0000000d-01e465bcbe69dbee-0ba2
Hash Value 3200 Valid
Handle to Object
Object Id 1,061,249
Object Hash 9829
File Id 1,061,249
File Hash 9829
Type File
Flags 0x0
Generation 8,215,264,866,564,785,059
Core Server Id 13
Freeze Refresh Dismiss

This window shows details about a file or directory.

Field descriptions

Path Name

The pathname to either the file or the directory.

Object Type

The type of object being displayed, either File or Directory.

Class of Service

The name of the Class of Service in which the file is stored. If the displayed object is a directory the value of this field will be `NONE`.

File Family

The name of the file family to which the file has been assigned. If the file has not been assigned to a family, the value of this field will be `Not in a family`.

Subsystem Name

The name of the HPSS subsystem which contains the file or directory.

Realm Id

The ID number which identifies the realm which encompasses the file or directory.

Account

The account index that is associated with the file or directory. This value is used by the HPSS accounting subsystem.

Read Count

This field applies only to files and is the number of read operations that have been issued against the file. Reading a file through most HPSS interfaces may cause multiple read operations to be issued, so this value may increase by more than one with each user read of a file.

Write Count

This field applies only to files and is the number of write operations that have been issued against the file. Writing a file through most HPSS interfaces may cause multiple write operations to be issued, so this value may increase by more than one with each user write of a file.

Link Count

For directories, this is the count of the number of directories that point to this directory. This includes "." (current directory) and ".." (parent directory).

Creation Time

Time and date that the file or directory was created.

Modify Time

Time and date that the metadata associated with the file or directory was last modified.

Last Written Time

This field only applies to files and is the time and date of the last write to the file. If this field is blank, the file has never been written.

Last Read Time

Time and date of the last read from the file, or the time and date of the last read from the directory.

Data Length

For directories, this is the byte length of the metadata associated with this directory. For files, this is the largest numbered byte in the file. Note that this is not necessarily the number of bytes in the file as files can be written sparsely with gaps. Also, it is not necessarily the highest written byte, since the Core Server supports POSIX `clear` and `truncate` commands which can leave a gap at the end of the file.

UID

The user ID of the owner of the file or directory.

GID

The group ID of the owner of the file or directory.

Permissions

Permissions granted to the file or directory. These are displayed in the standard UNIX mode format (read, write, execute permissions for user, group, and other). These are base permissions and do not take into account the effects of any ACLs on the file or directory.

Option Flags

Don't Purge

This field applies only to files. If the **Don't Purge** check box is selected, the file is purge locked and will not be purged from the top level of the hierarchy. If the **Don't Purge** check box is not selected, the file may be purged as usual.

Super Don't Purge

This field applies only to files. **Super Don't Purge** represents an system, non-user purge lock. It is usually associated with a recover operation. If **Super Don't Purge** is checked, it cannot be cleared except by cancelling the system operation which has set the flag (e.g. recover).

Purge On Migrate

This field applies only to files. When this field is selected, any time the file is migrated, it it immediately purged from a source disk level. This field does not take effect if the file is purge locked.

Extended ACL

If any ACL entry other than the default ACL entries exist, then the file or directory is said to contain extended ACLs. There are three type of ACLs that could have extended ACLs:

Object ACL

HPSS Name Space Object ACL

IC ACL

HPSS Name Space Initial Container ACL

IO ACL

HPSS Name Space Initial Object ACL

A check mark will be put into each of the ACLs containing extended ACL entries for this file or directory. For example, if a directory contains an USER ACL entry in the Object ACL and Initial Container ACL, then both the **Object ACL** and **IC ACL** check boxes will contain check marks.

Fileset Information

The Fileset Information group contains key fileset attributes for the root node of the fileset containing the file or directory.

Fileset Id

The ID number which identifies the fileset containing the file or directory . The 64-bit Fileset ID is displayed as two 32-bit unsigned integers separated by two commas.

Fileset Root Object Id

The ID number which identifies the root node of the fileset which contains the file or directory. This value is a 64-bit object ID.

Fileset State

Filesets have three access states: read, write, and destroyed. If a fileset allows reading, a check mark is displayed in the box labeled Read. If a fileset allows data to be written, a check mark is displayed in the box labeled **Write**. If a fileset is turned off, a check mark is displayed in the box labeled **Destroyed**.

Trashcan Information

When the HPSS trashcan feature is enabled, each HPSS object that is deleted causes a record (row) to be made in the NSTRASH database table. If the object that is displayed is currently in the trashcan, then the **Trashcan Information** fields will be filled in with the state of the object in the trash. If the object being displayed is not currently in the trashcan, then the **Trashcan Information** fields will be empty.

Parent Id

The Object ID of the Parent directory that contained the deleted entry.

User Id

User ID of the person who actually performed the deletion. (It isn't always the owner who performs the deletion.)

Time Deleted

The epoch time that the deletion was performed. This time is needed so that we can compute the length of time that an entry has been in a trashcan.

Time Created

At the time an entry is deleted this time is taken directly from the entry's object record. If this

entry is ever undeleted, this time will be restored back into the entry's object record.

Time Last Read

At the time an entry is deleted this time is taken directly from the entry's object record. If this entry is ever undeleted, this time will be restored back into the entry's object record.

Time Modified

At the time an entry is deleted this time is taken directly from the entry's object record. If this entry is ever undeleted, this time will be restored back into the entry's object record.

Original Pathname

The full pathname to the deleted entry. Note that Path is always relative to the root node of the fileset which contained the deleted entry. Path is useful when attempting to undelete an entry—it allows us to try to move the file back to its original location. It can also be useful when simply attempting to recall the original location of a particular trashcan entry.

Original Filename

This is the original entry name before the entry was moved into the trashcan. When entries are moved into trashcans their Names are purposely changed. This is done to prevent possible name conflicts. The Names are changed using the following format: <Name>.<ObjectId>. For example, if file `Foo` is deleted, its Name in the trashcan might be `Foo.12345`. Note that if an attempt is made to undelete `Foo.12345`, its name can be restored to its original Name, `Foo`.

Bitfile Id

If the object is a File, the bitfile ID to that file is displayed here. This **Bitfile Id** is used by the Trashcan Statistics Thread to obtain and report the length of all Files residing in trashcans. If the object is not a File, the **Bitfile Id** will not be displayed on the screen.

Object Id

Every bitfile is assigned a unique identifier.

Hash Value

The **Hash Value** is used to identify the database partition to which the bitfile is assigned. If the **Valid** check box is not set, the value of **Hash Value** is not available.

Valid

This check box is set if the value of **Hash Value** is available and not set otherwise.

Handle to Object

Object Id

Every file, directory, symlink, and other name space object is assigned a unique identifier. The **Object Id** field in this object handle contains this unique identifier.

Object Hash

The **Object Hash** is generated by running the object's **Parent Id** and the object's Name through a hashing algorithm. This hash value is then used to identify which database partition the object is assigned to.

File Id

File objects need an additional identifier when the file is a HardLink. If the File object is not a HardLink, the **File Id** is equal to the **Object Id**.

File Hash

The **File Id** and the **File Hash** fields are only used when the object identified by the **Object Id** field is a HardLink. When this is the case, the **File Hash** identifies the database partition that holds the metadata information needed by the HardLink.

Type

The **Type** of the object identified by this object handle. Valid object Types are File, Directory, Junction, SymLink, and HardLink.

Flags

If this object is the root node of a fileset, the NS_OH_FLAG_FILESET_ROOT bit is set in the **Flags** field.

Generation

The **Generation** field contains a uniqueifier. When verifying this object handle, this uniqueifier is compared to the **Generation** number kept in the object's metadata.

Core Server Id

This is the ID of the Core Server that issued this object handle.

Active Writes

Number of active writes against an open file. If the file is not open, the value is -1.

Active Reads

Number of active reads against an open file. If the file is not open, the value is -1.

Active Migrates

Number of active migration operations against an open file. If the file is not open, the value is -1.

Active Modify

Number of active modification operations against an open file. If the file is not open, the value is -1.

Active Copy

Number of active copy operations against an open file. If the file is not open, the value is -1.

Active Stages

Number of active stage operations against an open file. If the file is not open, the value is -1.

Freeze

Clicking on this check box freezes the window; in other words, all automatic updates to the window are suspended. Toggling the check box again returns the window, and the check box, to their normal conditions. Any accumulated changes that occurred while the window was frozen

are immediately displayed.

Buttons

Refresh

Requests that the current information about this fileset be fetched from the Core Server.

Dismiss

Closes the window.

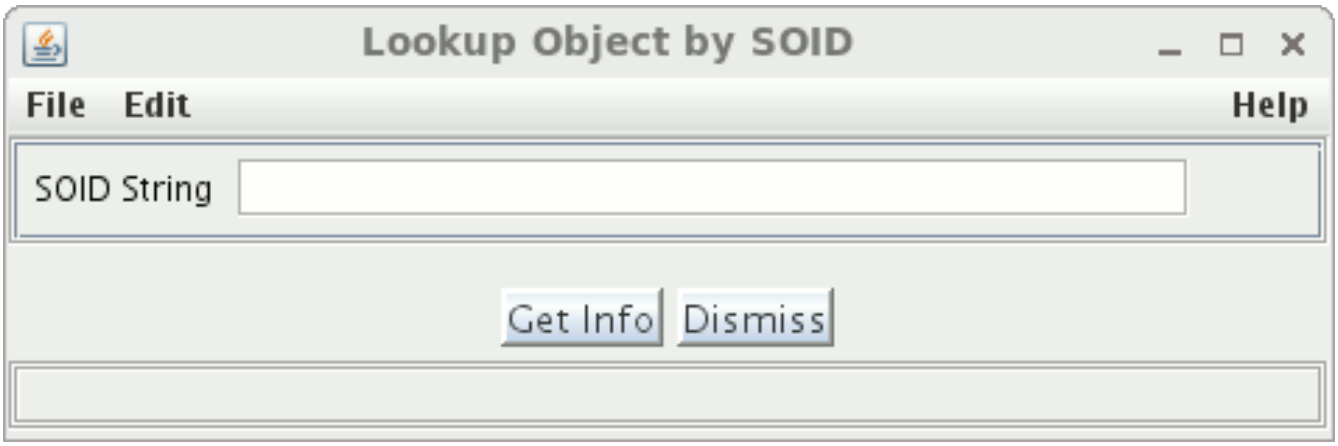
Note: The table below indicates which fields and attributes apply to both files and directories or to just one of the two. Refer to it to see which fields on the *File/Directory Information* window apply to files, directories, or both.

Table 31. Attributes of files and directories

Attribute	File	Dir
Account	x	x
BitfileId	x	
RealmId	x	x
Comment	x	x
CompositePerms	x	x
COSId	x	
DataLength	x	x
EntryCount		x
ExtendedACLs	x	x
FamilyId	x	
FilesetHandle		
FilesetId	x	x
FilesetRootId	x	x
FilesetStateFlags	x	x
FilesetType	x	x
GID	x	x
GroupPerms	x	x
LinkCount		x
ModePerms	x	x
Name	x	x
OpenCount	x	
OptionFlags	x	
OtherPerms	x	x

Attribute	File	Dir
ReadCount	x	
SubSystemId	x	x
TimeCreated	x	x
TimeLastRead	x	x
TimeLastWritten	x	x
TimeModified	x	x
Type	x	x
UID	x	x
UserPerms	x	x
WriteCount	x	

22.2. Objects by SOID window



To display this window, select **Monitor** from the *HPSS Health and Status* window, then select **Lookup HPSS Objects**, and then **Objects by SOID**. This window allows you to open an information window for an HPSS object which you specify by the object's HPSS Storage Object ID (SOID). The object types supported by this screen are Bitfiles and Virtual Volumes (Disk and Tape).

Field descriptions

SOID String

Storage Object ID string is the internal computer generated name for files, directories, and virtual volumes. HPSS utility programs that display SOIDs should present this field in the correct format.

Buttons

Get Info

Once all of the fields have been filled in, clicking on this button will open the information window for the object type you selected. The *Lookup Object by SOID* window will remain open so that another SOID may be specified.

Dismiss

Closes the window.

Chapter 23. Tape aggregation

This chapter discusses the following operations:

- Overview of tape aggregation
- Tape aggregation performance considerations
- Ordered Migration and Full Aggregate Recall
- Configuring tape aggregation
- Caveats

23.1. Overview of tape aggregation

Tape aggregation reduces file processing time when migrating relatively small files from disk to tape. In certain situations it may improve very-small-file disk-to-tape migration performance by two orders of magnitude over traditional, file-by-file migration. Tape aggregation also allows more files to be stored on a single tape volume by dramatically reducing the number of tape marks written to tape.

HPSS defaults to writing tape aggregates for all disk-to-tape migrations. If for some reason any file cannot be migrated to a tape aggregate, HPSS may instead migrate it as a single file. The maximum number of files in an aggregate is set to 50,000 files. This can be constrained using the HPSS_MAX_FILES_IN_AGGR environment variable in `env.conf`. Ensure the Core Server configuration is set correctly to accommodate the appropriate Core Server I/O limits. See Core Server I/O limit configuration for more detail.

In addition, the number of maximum open files configured for HPSS may need to be greater since the maximum number of files allowed per aggregate has increased. See Core Server max open files configuration for more information.

During single file migration, HPSS migrates each file into a separate tape segment. This generally causes the tape drive to flush the file data to the tape and subsequently write a tape mark. It is not unusual for the majority of the time spent migrating small files to be dominated by writing the tape mark.

Tape aggregation is enabled automatically in the disk migration policy, and if there are a sufficient number of migration candidate files ready to be migrated, HPSS will pack the data of many small files into a single tape storage segment. Thus, the 'end of file' tape mark is only written at the end of the batch of files, rather than one tape mark for each file being migrated.

Note that tape aggregation is managed internally by HPSS. It has no effect on file security nor on how files are accessed and generally has little effect on file retrieval performance.

23.2. Tape aggregation performance considerations

In order to get good performance with tape aggregation, there are several points to consider.

First, there needs to be a sufficient number of files ready to migrate. If migration runs too often, or if the HPSS system doesn't tend to ingest a large number of small files in a relatively short period of time, then there may be little or no performance benefit to tape aggregation since there will be very few files, if any, in each aggregate. Overall numbers of tape marks written to tape will still be reduced, but not as dramatically as in situations where there are relatively many files to migrate.

Second, tape aggregation performs best with relatively small files since most of the time spent to migrate them occurs during tape mark processing when the data is flushed to the media. From a data transfer rate performance perspective, it usually isn't useful to aggregate large files since the tape mark processing time is dominated by the time needed to write the file data. However, even with large files, the benefits of reduced tape mark usage still leads to storing more files on a tape volume.

For very large aggregate sizes, HPSS disk Movers may begin timing out during the migration. The Mover environment variable **MVR_CLIENT_TIMEOUT** may need to be increased on the Movers. See Additional Mover configuration for more detail.

The size of an aggregate is configurable by setting the "Max Aggregate Size" field in the disk migration policy. This value is an absolute value, and care must be taken to ensure a correct value if the migration policy is shared between storage classes.



Total Migration Stream Distinction

One migration stream consumes 2 times the "Max Aggregate Size" bytes before a secondary migration stream becomes active. These values will control the number of migrations streams being processed concurrently.

23.3. Ordered Migration and Full Aggregate Recall

The Ordered Migration feature orders files being migrated from disk to tape by directory or by create time. Further, for files being aggregated on tape, the tape aggregates can be used to group files according to their directories. This is especially pertinent when considering how to employ the Full Aggregate Recall feature.

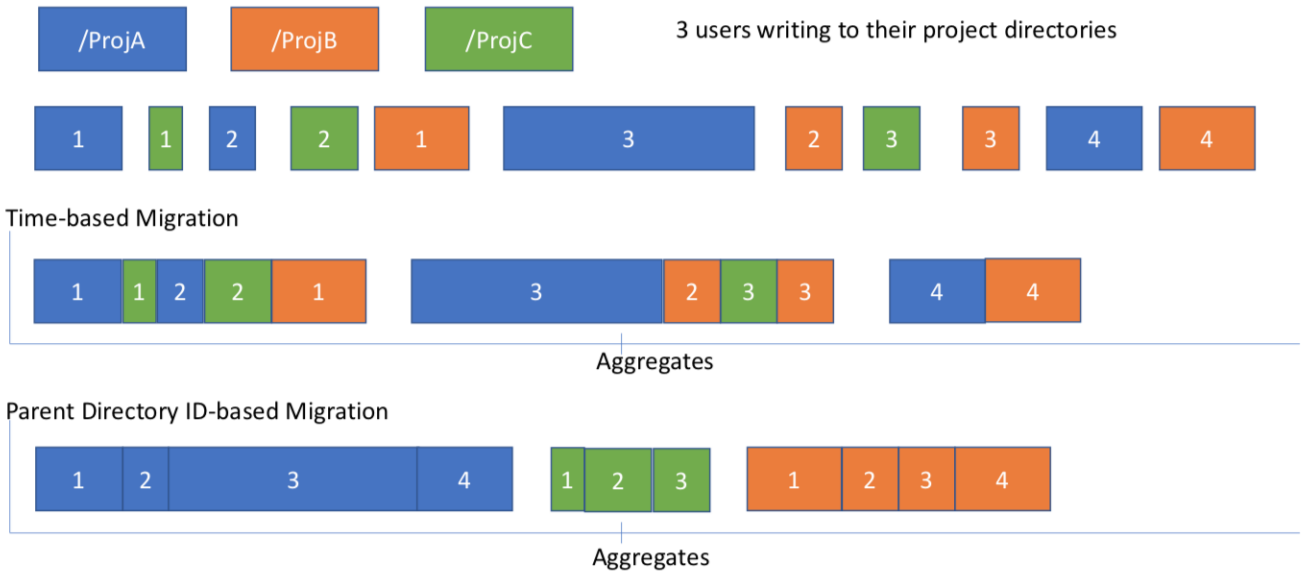
The objective of Ordered Migration is to co-locate data on tape, with either the directory of the files or the files' create times being the co-location criteria. A site may want to combine this feature with Full Aggregate Recall to potentially minimize the number of tape operations required to bring a directory's files back from tape. Of course, the migration policy will continue to honor the migration policy settings for when a file is eligible for migration and how often migration will run.

Ordered Migration also makes use of a **Disk Migration Policy** configuration option which specifies whether or not aggregates should be optimized for reading or writing. **Read-Optimized Aggregates** is defined as having only one ordering key value per aggregate (e.g. only files contained in Directory X). Select **Read-Optimized Aggregates** if you want this behavior. When **Read-Optimized Aggregates** is not selected, HPSS will create write-optimized aggregates: aggregates that may contain files from more than one ordering key value (e.g. files contained in directory X followed by files contained in directory Y, and so on). The is the default behavior. It is another important aspect to consider when preparing to employ Full Aggregate Recall.

Full Aggregate Recall allows an entire aggregate to be streamed from tape to disk. When this feature is on, and a stage is requested for a file which resides within a tape aggregate, then (apart from deleted files and already-staged data) the entire aggregate is staged to disk via a minimal number of tape I/O operations.

Migration Order and **Full Aggregate Recall** are configured on a per Class of Service basis. Refer to Class of Service Configuration window for details on configuring the Class of Service.

23.3.1. Ordered migration example - read-optimized aggregates



23.4. Configuring tape aggregation

Tape aggregation is turned on by default. To disable tape aggregation and migrate as single file, set the "Max Aggregate Size" field to 0 in the migration policy.

23.5. Debugging tape aggregation

If files are not being migrated into aggregates in the expected manner, it can be helpful to turn on batch migration debugging. To do this, add the environment variable HPSS_BATCH_MIGR_DEBUG_LEVEL=1 to the `/var/hpss/etc/env.conf` file. Then restart or reinitalize the Core Server to make it notice the new environment variable.

The Core Server will then issue additional DEBUG level log messages when certain aggregation errors are encountered, such as a file being rejected from a batch. These log messages will include the string "BATCH-FILE-DEBUG."

To stop the extra logging, remove the definition of the environment variable from the `env.conf` file, or reset it to 0. Then restart or reinitialize the Core Server.

23.6. Caveats

Tape-to-tape migration of files in aggregates is an unsupported configuration. If multiple copies of

aggregated files are needed:

- Set the "Max Aggregate Size" field to 0 in the parent disk storage class's migration policy, and
- Configure the corresponding hierarchy for migration from the disk storage class to two or more tape storage classes.

Chapter 24. Tape Ordered Recall (TOR)

This chapter discusses various aspects of HPSS tape ordered recall including high level concepts, configuration, and monitoring.

24.1. Overview of TOR

Tape Ordered Recall reduces file processing time when recalling data from tape. In certain situations it may improve tape recall rates by one to two orders of magnitude over unordered access. TOR works with or without the Recommended Access Order feature.

TOR is an HPSS feature that implements scheduling, queueing, and reordering of tape requests. If there are multiple tape read requests for the same virtual volume in HPSS, those requests will be queued up in the HPSS Core Server. That queue will then be ordered before the next request is pulled off to be worked, so that the requests are worked using the queueing criteria specified in the *Core Server Configuration* window.

Tape does not deal well with random access. Due to the nature of the media, seek times are generally measured in seconds. This makes unordered tape I/O particularly poor performing, especially when the random accesses involve relatively small files.

TOR attempts to reduce tape seek times by issuing tape requests after they have been ordered. TOR is always on, but has a number of configuration options which can impact its behavior and HPSS tape recall performance.

There are a number of APIs which implement batch staging. These APIs are described in detail in the *HPSS Programmer's Reference*. They include support for issuing multiple synchronous or asynchronous stage requests with a single API call and retrieving schedule status along with stage status data. Additionally, the quaid(7) tool can take advantage of these APIs to facilitate mass recall from tape.

Additionally, the lori(7) tool makes use of ordered recall APIs similar to staging. The lori(7) tool takes advantage of the read queueing APIs.

24.2. Multiple User Considerations

Both the TOR feature and the HPSS Request Queue that is built on top of it are designed for multi-user access. Any idea of a queue for a volume will always be a **global** queue across all users, where all user requests are optimized. So for instance if multiple users submit a batch of requests, each batch is broken apart according to volume and the requests are scheduled against the individual tapes on a global basis (that is, all user requests will be intermingled to optimize the use of the tape).

HPSS has historically treated requests from all users with equal priority in gaining access to tape. In the future, HPSS may allow for prioritization.

24.3. TOR ordering behavior

There are two queuing methods available with TOR: **FIFO** and **Offset**. These are described in detail below.

FIFO

FIFO will issue incoming requests in the order they were received by the Core Server tape subsystem. This generally reflects the order that clients submitted the requests.

Offset

Offset orders the requests using offset information on the volume. The mechanism for this can differ depending upon the tape drive that the volume is mounted in. For drives which support it, the Recommended Access Order (RAO) will be used. This feature allows the tape drive to inform the application of the best order to read a sequence of requests from that particular tape. If the volume is mounted in a drive without RAO support, the Core Server will order the requests in linear tape order, beginning with the current position of the tape.

Other aspects of TOR behavior

1. Ordering occurs when the tape is idle, after a request has come into the scheduler and has not been ordered.
2. Ordering requires that the tape be mounted in order to support the RAO feature.
3. When the first request enter a tape volume queue, HPSS will delay selecting it to run for a few seconds to allow other requests to enter the queue. If no other requests enter the queue, then the request will begin to run. Otherwise, HPSS will wait a little longer until new requests for the volume stop appearing before ordering the requests and beginning to work through them. This allows HPSS to optimize an incoming batch of requests.

24.4. TOR performance considerations

In order to get good performance with Tape Ordered Recall, there are several points to consider.

There need to be a sufficient number of parallel stage or read requests coming in from the application for data on tape. If tape read requests come in a handful at a time there may be little to no gain seen with TOR. TOR can only schedule what it knows about, so it is important to provide a steady stream of parallel read requests into HPSS so that TOR may schedule them. Since TOR works at the virtual volume level, it is most beneficial if these parallel reads are for data on a small set of volumes. Direct read from tape, synchronous stages, and asynchronous stages all go through the TOR queuing logic, so the type of read request doesn't impact whether requests will be ordered. More read or stage requests queued up allows for better scheduling, which should result in less seek time and improved tape recall performance.

Parallel access simply means spawning a number of clients reading or staging from the same tape volume at the same time. Direct reads from tape, synchronous stages, and asynchronous stages all go through the TOR queuing logic; the type of tape read access doesn't impact whether requests will be ordered. More read requests queued up allows for better scheduling, which should result in less seek time and improved tape recall performance. This means that general client access which interacts with tape will be automatically optimized against other work on the same tape volume.

However, this method runs into resource limitations at the scale (described below) so it is not recommended for very large batch work.

The simplest way to use the batch read or stage interfaces at scale is to use **lori** for reads and **quaid** for stages. This allows a single client to submit a large number of requests to HPSS without the resource utilization or coordination requirements of a large number of parallel client requests. These batch tools can accept file input lists of thousands or tens of thousands of read or stage requests and optimize the access to tape automatically. See Using Quaid and Lori for more information. Application developers can implement this in their local applications by using the APIs described in the HPSS Programmers Reference.

24.4.1. TOR General Configuration Parameters

The following configuration parameters are important in both the parallel access and batch access models.

Maximum Active I/O Requests

The Core Server I/O scheduler has a limit to the number of I/O requests that are processed, subsystem-wide. For the parallel I/O method described above, each read request that triggers a stage counts as a single I/O. In order for a sufficient number of parallel tape read requests to get passed down to the tape scheduler, it may be necessary to increase this value. For example, in order for the system to process 100 tape read requests in parallel, this value would need to be at least "100". For the batch model, the Maximum Active I/O Requests should be greater than or equal to the Max Request Queue Worker setting - each worker can potentially be handling an I/O request.

Maximum Active Copy Requests

The Core Server I/O scheduler has a limit to the number of data copy requests that are processed, subsystem-wide. Each stage request, including each individual file within a batch stage request, counts as a single copy. Like *Maximum Active I/O Requests*, this value needs to be configured appropriately for the expected stage workload. For example, in order for the system to process 100 stage requests, this value would need to be at least "100".

Tape Queue Method

The Core Server's Queue Method has a major impact upon performance with TOR. The default is *Offset*; the Core Server will use RAO for tapes volumes mounted in suitable drives and linear tape ordering for all others. Using *FIFO* should only be done if the site is already ordering all requests from tape.

Tape Queue Active Set Size

The Core Server's Active Set Size controls the number of stage requests, per volume, which are scheduled together. For example, if a volume has 1000 pending stage requests, and the *Active Set Size* is set to "500", then the first 500 requests to enter the queue will be ordered and run. Then the next 500 requests will be ordered and run. Lower values can impact performance because it limits the requests that the scheduler can consider for generating a well-performing I/O schedule. Higher values can increase the average wait time of individual requests into the scheduler.

Level 0 Disk Storage Class Max Segment Size

For stage requests in a disk over tape hierarchy, asynchronous stage requests will be done in chunks based on the disk storage class *Max Segment Size*. Increasing the disk *Max Segment Size* will result in larger read requests from tape, which will result in better performance. Synchronous stage requests are unaffected. For example, to asynchronously stage a 1000 MB file up to a disk storage class with a 100 MB max segment size, it would require 10 reads from tape.

24.4.2. Mass Parallel Access Configuration Parameters

The mass parallel access model, described above, is a situation where a site wants to optimize access to HPSS via a large number of concurrent client requests. Some situations where this access model would occur are:

1. Client applications cannot take advantage of batch APIs
2. Large number of concurrent clients from different sources

The upper bounds of the mass parallel access model are defined by the number of open sockets on the HPSS core server system - usually in the realm of tens of thousands of concurrent requests.

HPSS has a number of configuration options which can impact the performance and behavior of TOR in the parallel access model. These are described below:

Maximum Connections

The Gatekeeper (if configured), and Core Server maximum connections will limit the number of client connections, which could limit incoming tape read requests. This parameter should be set to a value which will allow the desired throughput of tape read requests to the server.

Gatekeeper Settings

For sites which make use of a Gatekeeper and a site policy which restricts opens or stages, there may be adjustments to make to allow additional requests to come through as part of a large set of stages against a single VV. These adjustments may differ depending upon the implementation of your site policy.

Thread Pool Size

Same as above.

Max Open Files

In the mass parallel access model, a client will open all files being staged or read in parallel prior to reading. This can be a significant number of open files that are waiting to be read. The Core Server should have a sufficient number of open files to handle the parallel read load as well as other system activity.

Max Open System Files and Sockets

The system itself must allow for a large number of open files and a large availability of sockets for clients to connect to. For example, the socket range for a core server system should always be greater than the Maximum Active I/O Requests setting.

24.4.3. Batch Access Configuration Parameters

Read and stage batching involves an application submitting a number of requests together to HPSS. HPSS will manage how the requests are run and report back when the stage is complete or the read is ready to run. The advantage of this model is that it altogether consumes far fewer resources in terms of open files, network sockets, etc. Some situations where this access model would occur are:

1. In the general use of many provided HPSS applications, for example GHI, quaid or lori
2. The site has or is building a workflow management system for mass recall or stage based on user batch recall submissions
3. The administrator has ad-hoc, periodic tasks that involve mass recall or stage operations that need to be optimized

HPSS has a number of configuration options which can impact the performance and behavior of TOR in the batch access model. These are described below:

Maximum RQ Request Threads

Core Server field which controls the number of worker threads in the core server dedicated to processing request queue entries for read and stage callback requests. This should be set to at least the number of tape drives that should be used for recalls and stages in the system.

Maximum RQ Discovery Threads

Core Server field which controls the number of threads in the core server dedicated to retrieving information about read and stage callback requests. This information is required for optimizing access to tape. This should be at least 10% of the request thread count.

24.5. Configuring TOR

RAO-enabled tape drives may have their RAO capability enabled or disabled in SSM using the tape drive configuration screen. See Configure a new device and drive. This provides the finest granularity capability for enabling or disabling RAO per tape drive type. Drives which have RAO disabled will use offset-based ordering unless FIFO is selected for the volume.

There are three additional methods to modify TOR behavior, even while the system is running, without a Core Server reinitialization. These are meant to be used in diagnostic or temporary error situations and are described below.

Globally Disable Scheduling

By touching a file named `$HPSS_TMP/noschedule` (for example, `/var/hpss/tmp/noschedule`), any Core Servers which can read that file will disable all reordering of requests. Requests will still be queued, but they will not be reordered until the next opportunity after the file is removed.

Globally Disable Recommended Access Order

By touching a file named `$HPSS_TMP/nodrive` (for example, `/var/hpss/tmp/nodrive`), any Core Servers which can read that file will not use the Recommended Access Order (RAO) feature, but will instead do linear offset ordering. Upon removing the file, ordering with RAO will be done the next time the queue would be reordered.

Mark an individual volume as FIFO

Individual volumes can be configured to be read in a FIFO manner via the *Core Server Tape Volume* information window. This can assist in reading certain types of damaged volumes.

24.6. Monitoring Tape Order Recall

Tape Ordered Recall writes a periodic dump file, one per subsystem. This file is dumped roughly every five seconds and contains the current schedule, and statistics in JSON format. This can provide a site with data and metrics regarding requests, including their order within the queue, their length, and how long they have been waiting.

This file is located in `$HPSS_TMP/tape_schedule_<subsystem-id>.json` (for example, `/var/hpss/tmp/tape_schedule-1.json`).

A tool, **torsummary**, can read this file and present the information in the terminal. This can be useful for monitoring both pending read and write activity against tape volumes.

24.7. Caveats

1. Recommended Access Order currently has a hardware limit of 2730 requests that can be ordered. Sites using RAO should be aware that while they can set a larger Active Set Size, the scheduler will cap the number of requests that will be ordered via to the drive-specified limit of 2730.
2. While RAO is a useful feature, there are diminishing returns as more files are organized on a tape at the same time. There is often only a minor difference in submitting a batch stage of 1000 files vs 100 files. In other words, the benefits of RAO are significant in ordering 100 files vs a sorted retrieval, but almost unnoticeable in ordering 100 files at a time with RAO vs 1000 files with RAO. Sites should not go to extremes to push large batches of requests into the system since the performance benefits are relatively small vs submitting smaller batches. Stage and read persistence and recoverability may be good reasons for submitting larger batches of requests to the system.

24.8. HPSS Core Server Request Queue

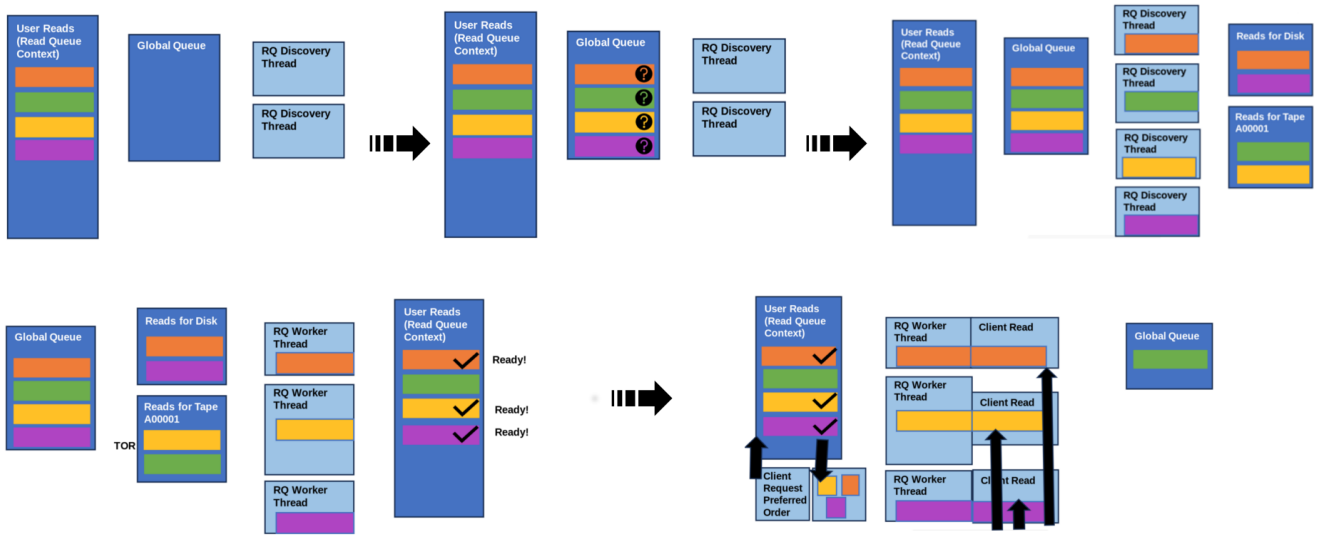
HPSS has a queue for managing client I/O operations at scale in the Core Server. This queue organizes read and stage operations which come in through the batch tools. These tools come into the request queue and are persisted, organized, and executed by a limited number of request queue threads.

The request queue splits requests up by volume, with all disk requests being treated equally. For tape volumes, each worker thread will identify one tape volume to work on and begin pulling requests for that volume until there are no more requests. Any remaining workers will work off requests for files on disk.

For stage activity, the stages are issued by the workers and issues any necessary callback when the stage completes, and there is no dependence on the client after they have submitted the request.

For read activity, the situation is more complex. When the core server request queue processes a read request, that read request is marked as "ready" and the (polling) client is expected to notice it is ready and issue the read request. This results in the core server request queue processing linking up with the client read request and the read is executed as normal. However, in the case where the client cannot issue the read or does not remove the request from the queue within 3 minutes, the client with the queued reads gets a strike. When the client has three strikes, their further queued requests will be aborted to allow access for other clients. Thus, it is vital that applications that use the read queuing APIs implement their polling and read requests in a way where reads that have come "ready" are being issued quickly or removed otherwise.

Below is an example of how HPSS read queue requests work.



In this example, starting at the top left, we can see a system with no requests in it and four user requests coming in represented by the different color rectangles. The global list is empty and the discovery threads are not doing anything. In the next figure to the right, we see that the user's reads have been placed into the global queue, but we don't know what volumes they correspond to. The discovery threads will identify that these requests need to be organized, and will find out which volume they should go in. This brings us to the next figure, where the requests have been sorted. Two of the requests are for a disk volume and two are for a tape (A00001).

On the second row, we can see that the requests for tape A00001 have been organized using TOR, and then worker threads have come in to pick up three of the four requests to initiate work. The fourth request (green) is not picked up because it is on tape, and only one read request against tape can occur at a time. Also in this picture, we see that orange, yellow, and purple are now "ready" because there is a read thread working on them.

In the last image, we see the client has been notified that these three requests are ready to be read. The client issues its three parallel reads for those requests, which match up with the associated worker threads. Finally, we still have green waiting to be read since it is on the same tape as yellow. It will be read after yellow finishes, unless another request for A00001 comes in which is more optimal to read.

24.8.1. HPSS Core Server Request Queue Monitoring

There are two ways to monitor the HPSS Request Queue. The first is by doing a SIGHUP on the Core Server. The status of the queue will be dumped in the normal SIGHUP output. This output is the live, in-memory state of the queue.

The second is by using the **lsqueue** tool (see man page for details). This method dumps the request queue information that is persisted in db2 rather than the live state, but is still useful for querying activity.

24.8.2. HPSS Core Server Request Queue Management

HPSS Core Server Request Queue management takes two forms: aborting requests and clearing persisted requests.

Aborting Request Queue Items

The **hpss_abort_io** tool can be used to abort stages and reads which are in the core server request queue. Additionally, the Client APIs which initiate abort requests can be used by client applications to abort items in the core server request queue.

Clearing Persisted Requests

The **rq_cleanup** tool can be used to remove metadata for stages or read requests. This tool is meant to be used during a down window to clear out requests which are no longer desired. By default, read operations will be ignored if they are over 24 hours old when the core server starts up and reloads them. Stages have no time limit. See the man page for usage information.

24.8.3. Using Quaid and Lori

Quaid and Lori are tools for batch staging and read operations, respectively. These two tools behave very similarly and share a common code base, with quaid implementing the batch stage interface and lori implementing the batch read interface. Both have extensive man pages which describe their use.



A note about reading vs staging. Staging operations are meant to be used to get data ready for an application to read from HPSS at a specified time, or to facilitate applications with workflows that do not work well with high latency storage (for example, needing to read small pieces of very large files multiple times during the process). The admin, when choosing whether to use quaid or lori, should consider whether this is a stage use case or if the data simply needs to be copied to a file system available to the lori client for use by an application. Staging is an inefficient extra data movement if the data can simply be read out.

Important options for quaid and lori:

- For large lists of files, using a bitfile list, **--bitfile-list** / **-B** is the fastest method. This causes the scanning stage to complete faster.
- The **--dry-run** / **-R** option does a dry run, which does the scanning and grouping step only. This can be useful to understand the stage or read workload that will be created by the tool when

staging directories recursively, for example.

- The **--complete-callback** / **-C** option allows for a callback executable to be provided to the tool. The tool will call that executable with information about each completed request. This is useful for building workflows around quaid and lori.
- Lori can be restarted by using the **-x** option if the tool stops unexpectedly. This can be done from another client, if necessary.

Generally, the performance options for quaid and lori should not be modified. For best results, run quaid and lori when there are a number of stages or recalls on a defined set of tapes. Consider that for quaid and lori, it is possible to consume all the tape drives with stages and recalls if the list of files spans enough tape drives unless the drive recall quota or HPSS core server request queue worker counts would not allow this. Consider the tape drive requirements of other activity (repacks, migrations, verification) when configuring the HPSS request queue or using quaid and lori.

See Batch Access Configuration Parameters for more information.

Chapter 25. User accounts and accounting

25.1. Managing HPSS users

After the HPSS system is up and running, the administrator must create the necessary accounts for the HPSS users. For a new HPSS user, a Kerberos, LDAP, or UNIX ID (depending on authentication type configured) must exist before the user can access HPSS. In addition, if the HPSS user needs to use SSM, an SSM ID must also be created before the user can use SSM. The SSM ID should be created only for the HPSS administrators and operators.

The HPSS User Management Utility (hpssuser(7)) provided with HPSS can be used by the administrator to add, delete, and list the HPSS user IDs. The utility must run as root to acquire the necessary authority to create new KRB, LDAP, UNIX, and SSM IDs. Refer to the hpssuser(7) man page for more information.

25.1.1. Adding HPSS users

The hpssuser(7) utility can be used by the administrator to add a UNIX user ID, a KRB user ID, an LDAP user ID, and an SSM user ID to the HPSS if these IDs do not already exist. The hpssuser(7) utility can be invoked to simultaneously add multiple types of user IDs for a user, or to add an individual user type.



Ensure that the Core Server is up and running before adding the user ID. The hpssuser(7) utility will not be able to create the user's home directory if the Core Server is not available.

25.1.1.1. Add all user ID types

The utility can be used to simultaneously add all relevant user ID types for a user. When invoked with the **-all** option, the hpssuser(7) utility will consult the system authentication and authorization configuration and add user types which are consistent with the configuration. For example, if the system is configured to use Kerberos authentication and LDAP authorization, a KRB user ID and an LDAP user ID will be added in addition to the UNIX user ID, and SSM user ID types.

Invoke the hpssuser(7) utility as follows to add the required user ID for an HPSS user:

```
hpssuser -add <user> -all
```

When invoked, the hpssuser(7) utility will prompt the user for any required data. The following is an example of adding all user ID types on a system configured to use Kerberos authentication and LDAP authorization. The **-nohome** option indicates that no user home directory will be created in HPSS.

```
# hpssuser -add user1 -all -nohome
```

```
User ID#: 300
Primary group name: hpss
Enter password for user1: ******
Re-enter password to verify: ******
Full name: Test User
Login shell: /bin/ksh
Unix (local/system) home directory: /home/user1
[ adding unix user ]
[ added unix user ]
[ KADMIN_PRINC unset; using kadmin.local for Kerberos ops ]
[ adding kerberos principal ]
[ added kerberos principal ]
HPSS/LDAP home directory: /
Primary group ID#: 210
[ adding ldap principal ]
[ added ldap principal ]
[ adding ssm user ]
1) admin
2) operator
Choose SSM security level
(type a number or RETURN to cancel):
> 1
[ ssm user added : admin ]
```

25.1.1.2. Add a user ID

The hpssuser(7) utility can be used to create HPSS user identities and create keytabs by using the following syntax:

```
hpssuser -add <user> -acct [-keytab <path>]
```

The utility will prompt the user for required data for the configuration. Following is an example of adding a UNIX user ID and creating a UNIX keytab:

```
# hpssuser -add user1 -acct -keytab user1.keytab
User ID#: 300
Primary group name: hpss
Enter password for user1: ******
Re-enter password to verify: ******
Full name: Test User
Login shell: /bin/ksh
Unix (local/system) home directory: /home/user1
[ adding unix user ]
[ added unix user ]
[ adding unix keytab entry to 'user1.keytab' ]
[ added unix keytab entry to 'user1.keytab' ]
```

Ensure that the Core Server is up and running before adding the user ID. The hpssuser(7) utility will

not be able to create the user's home directory if the Core Server is not available.

25.1.1.3. Add an SSM user ID

The hpssuser(7) utility creates an SSM ID for a user by adding an entry for him to the AUTHZACL table. Refer to SSM user security for more information on SSM user security.

Invoke the hpssuser(7) utility as follows to add an SSM user ID:

```
hpssuser -add <user> -ssm
```

The utility will prompt the user for the required data. Following is an example of adding an SSM user ID:

```
# hpssuser -add user1 -ssm
[ adding ssm user ]
1) admin
2) operator
Choose SSM security level
(type a number or RETURN to cancel):
> 1
[ ssm user added : admin ]
```

25.1.2. Deleting HPSS users

The hpssuser(7) utility can be used by the administrator to delete existing user IDs for an HPSS user. The utility can be invoked to delete all user IDs for the user or to delete an individual ID.

The utility will prompt the user for the required data. Following is an example of deleting the user IDs for an HPSS user on a system configured to use Kerberos authentication and LDAP authorization:

```
# hpssuser -del user1 -all
[ deleting ssm user ]
[ SSM user deleted ]
[ deleting ldap principal ]
[ deleted ldap principal ]
[ deleting kerberos principal ]
[ KADMIN_PRINC unset; using kadmin.local for Kerberos ops ]
[ deleted kerberos principal ]
[ deleting unix user ]
```

25.1.3. Listing HPSS users

The hpssuser(7) utility can be used by the administrator to list all existing HPSS user IDs. The utility can be invoked to list all HPSS user IDs or a particular type of user ID.

Following is an example of listing the *user1* user ID on a system configured to use UNIX authentication and authorization.

```
# hpssuser -list user1 -all
[ Kerberos not configured - skipping... ]
[ LDAP not configured - skipping... ]
[ unix(linux) user info ]
uid=300
gid=210
home=/home/user1
shell=/bin/ksh
fullname=Test User
[ ssm user info ]
(admin -> perms = 'rwxcidt')
(operator -> perms = 'r--c--t')
SSM client security ACL entry:
rwxcidt - user - 300 (user1) - 10000 (HPSS_REALM.NAME)
```



Listing all UNIX users is not supported.

25.1.4. Create an SSM client package

The hpssuser(7) utility can be used by the administrator to generate an SSM client package (all files required to start an SSM client on another node). The utility also has the ability to include HPSS documentation in the package if the **-packagedoc** option is specified. The following is an example of creating an SSM client package without the HPSS documentation on a system configured to use Kerberos authentication. For the case of a system configured to use UNIX, then `krb5.conf` won't be included in the package. Note that if `krb5.conf` does not exist on a system that's configured to use krb auth, then the command will fail.

```
# hpssuser -ssmclientpkg /tmp/ssmclientpkg.tar
[ packaging ssm client ]
[ creating /tmp/ssmclientpkg.tar ]
ssm.conf
krb5.conf
hpssgui.pl
hpssgui.vbs
hpss.jar
[ packaged ssm client in /tmp/ssmclientpkg.tar ]
```

25.2. Accounting

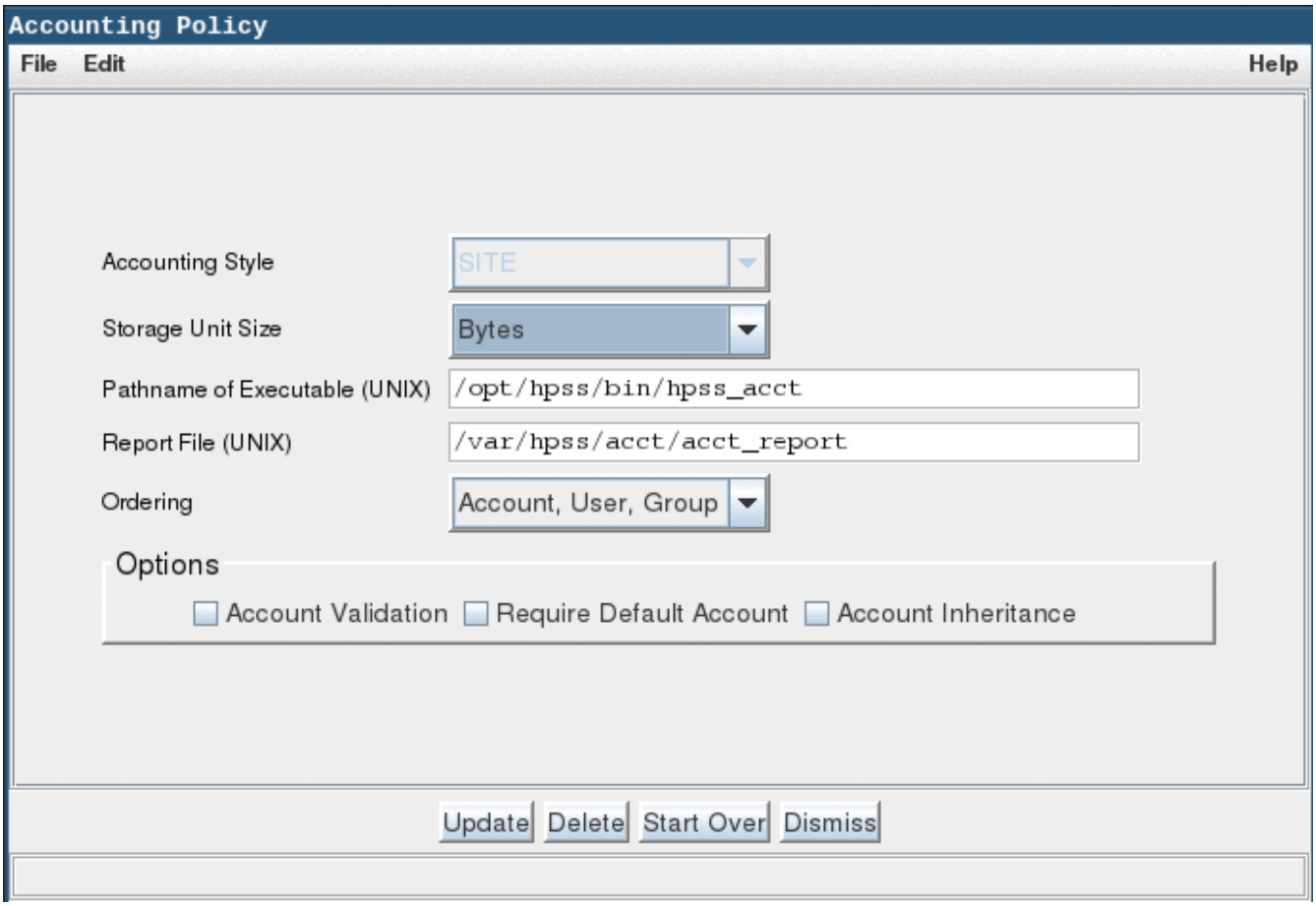
HPSS maintains accounting information on the usage of the system whether the site charges for usage or not. Sites are encouraged to use the accounting information, even if they do not charge users, to gain a better understanding of the usage patterns of their system.

The accounting policy defines how the usage of HPSS resources is reported. An accounting policy is

required whether the site actually charges users for HPSS usage or not.

The accounting policy may be created using the *Accounting Policy* window (select this window by clicking **Accounting** on the **Policies** submenu of the **Configure** menu on the *HPSS Health and Status* window). After the accounting policy is created, it can be viewed, updated, or deleted through the same window. Note, however, that once the accounting style is chosen, it may not be modified.

25.2.1. Accounting Policy window



This window allows an administrator to manage the accounting policy. If the **Update** button is visible, then the accounting policy already exists; otherwise, an **Add** button will be displayed. If changes are made to the **Storage Unit Size**, **Pathname of Executable (UNIX)**, **Report File (UNIX)**, or **Ordering** fields, after clicking **Update** the changes will take effect on the next generation of an accounting report. If any of the **Options** flags are changed, clicking the **Update** button and then clicking **Apply Config** on the *HPSS Health and Status* window will reinitialize the Core and Gatekeeper servers to allow the changes to take effect.

An accounting policy is required whether the site actually charges users for HPSS usage or not.

Field descriptions

Accounting Style

The style of accounting that is used by the entire HPSS system. Valid values are SITE or UNIX. The default value is UNIX.

Under UNIX-style accounting, resource usage is reported by user ID (UID). Each user is allowed to use only one account ID, which has the same numerical value as his user ID. Under SITE-style accounting, each user may use multiple account IDs.

Note that if you use SITE-style accounting, you must either enable **Account Validation** in the accounting policy or select LDAP as your authorization mechanism. See below for a description of the **Account Validation** field in the accounting policy. See Security services for a description of LDAP and other HPSS authorization mechanisms. See the Accounting policy and validation section for a complete description of accounting styles and account validation.



Advice

This field must be set up properly before any files are written to the system. Under UNIX-style accounting, HPSS obtains the account number from the user's UNIX UID. SITE-style accounting allows the account ID to be set independently of the UNIX UID. Once the accounting policy is configured, the accounting style cannot be changed.

Storage Unit Size

The integral units to be displayed in the report file. The valid values for this field are Bytes, Kilobytes, Megabytes, and Gigabytes. The default value is Bytes.

Pathname of Executable (UNIX)

The UNIX path name of the accounting utility executable. This field may contain any valid UNIX path name. The default value is "`/opt/hpss/bin/hpss_acct`".

Report File (UNIX)

The UNIX pathname where the generated accounting report will be stored. The Report File (UNIX) field can contain any valid UNIX pathname. The default value is "`/var/hpss/acct/acct_report`".

Ordering

Accounting may report users based on their Account ID, User ID, and/or Group ID. This field specifies which of the identifiers to use when creating an accounting report as well as the order of priority for sorting. For example, when selecting "`User, Group`", users will be sorted by User ID first followed by Group ID second. Any users that are identical with respect to the chosen identifiers are combined into a single row. For example, if there are two users `(Account ID = 1, User ID = 2, Group ID = 3)` and `(Account ID = 2, User ID = 2, Group ID = 3)`. If the ordering is set to "`User, Group`", the information about these two users are combined as they are the same user with respect to the chosen report ordering.

Options

Account Validation

A flag that indicates whether or not authorization of user accounts should be performed by a Gatekeeper. The default value is OFF.



Advice

If you turn this flag ON, you must configure at least one Gatekeeper which

will perform the account validation service.

Require Default Account

A flag that indicates whether or not users must have a valid default account index before they are allowed to perform any operation. The default value is OFF. It is only used if Account Validation has been enabled. If this flag is disabled, validation will occur only during file manipulation operations.

Account Inheritance

A flag that indicates whether or not newly created files and directories should automatically inherit the account index used by their parent directory. The default value is OFF. It is only used if Account Validation has been enabled and SITE-style accounting has been selected. If this flag is disabled, new files and directories have the user's current session account index applied to them.

25.2.2. Accounting reports and status

25.2.2.1. Generating an accounting report

The HPSS accounting report tool accumulates accounting information from metadata and generates a report file. The accounting report will include:

- Total file accesses, total number of files, and total space used per account per Class of Service.
- Total file accesses and amount of data transferred per account per Class of Service per storage class.

The accounting report represents a blurred snapshot of the system storage usage as it existed during the accounting run; that is, the accounting information is gathered while the system runs, so the accounting data is changing underneath it, and it may not represent what would be collected if no users were active during the accounting process. Accounting information is maintained per storage subsystem and the accounting reports summarize information for that storage subsystem.

Before creating an accounting report, you must create and fully configure an accounting policy. The initial accounting policy should be set up before any files are created in the HPSS system. Except for the accounting style, the accounting policy fields can be updated any time after the initial setup.

There are two ways to generate an accounting report: from the *Subsystems* window or from the *Accounting Status* window. For each method, begin by selecting the **Accounting Report** menu item from the **Operations** menu on the *HPSS Health and Status* window. This will bring up the *Subsystems* window. Click on the storage subsystem for which an accounting report will be generated. This will highlight the **Accounting Status** and **Start Accounting** buttons.

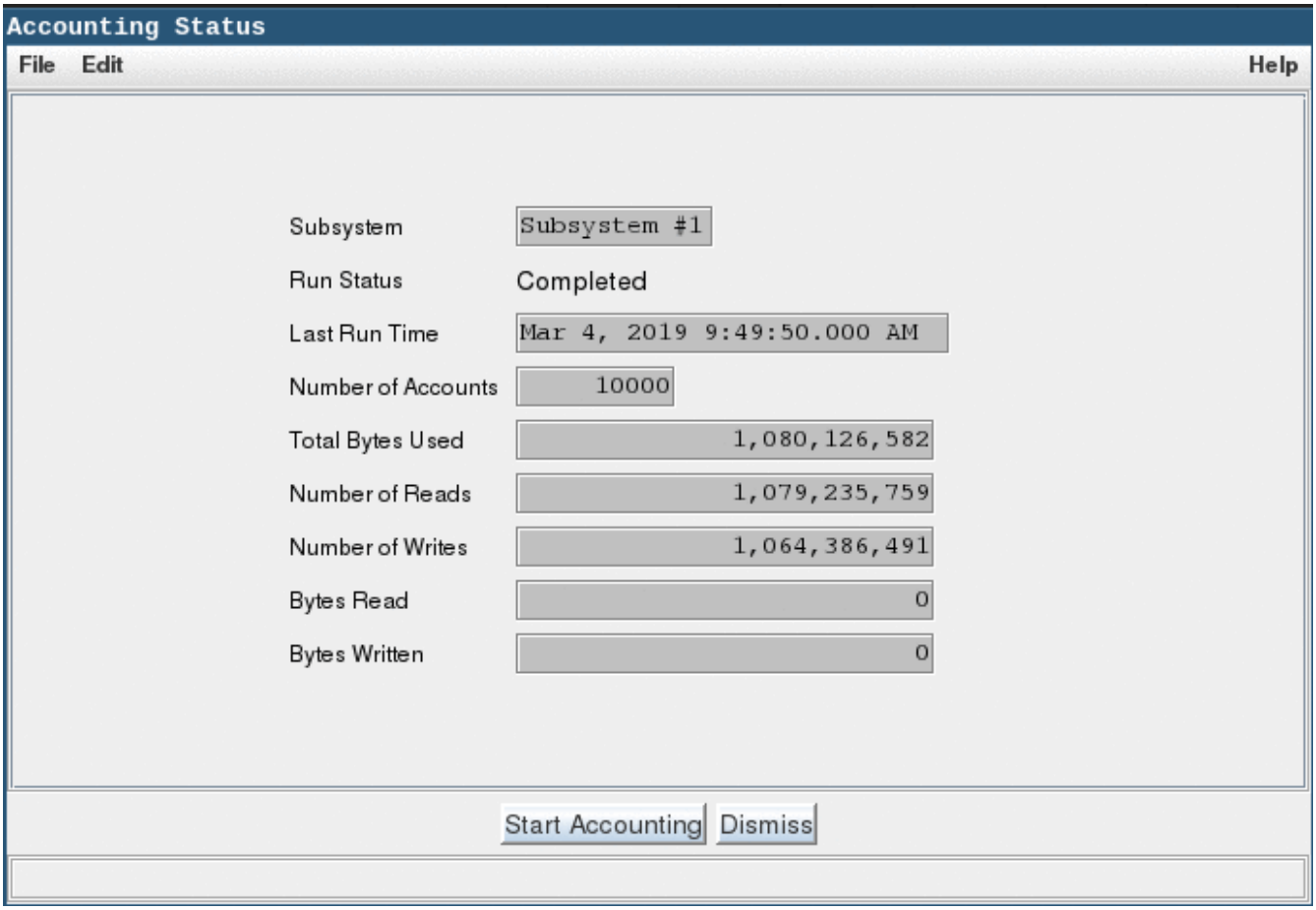
To create an accounting report, choose either of these methods:

- On the *Subsystems* window, click the **Start Accounting** button. The accounting report should run quickly and a completion status will appear at the bottom of the *Subsystems* window.
- Bring up the *Accounting Status* window by clicking on the **Accounting Status** button on the *Subsystems* window. On the *Accounting Status* window, click the **Start Accounting** button. The

accounting report should run quickly and a completion status will appear at the bottom of the *Accounting Status* window.

For either method, the *Accounting Status* window will display the overall statistics from the last accounting run.

25.2.2.2. Accounting Status window



This window allows an administrator to view the accounting status and start accounting.

Field descriptions

Subsystem

The name of the storage subsystem containing this accounting status data.

Run Status

Current status of accounting run. Possible values are:

- `Never run`
- `Running`
- `Failed`
- `Completed`
- `Report generated`

Last Run Time

If accounting is currently running, this is the time the run started; otherwise, it is the time the last run completed.

Number of Accounts

Total number of accounts in the storage subsystem. Set after a successful run.

Total Bytes Used

Total number of bytes accounted for in the storage subsystem. Set after a successful run.

Number of Reads

Total number of reads performed in the storage subsystem during the accounting period covered by the last successful accounting run. Set after a successful run.

Number of Writes

Total number of writes performed in the storage subsystem during the accounting period covered by the last successful accounting run. Set after a successful run.

Bytes Read

Total bytes read from the storage subsystem during the accounting period covered by the last successful accounting run. Set after a successful run.

Bytes Written

Total bytes written to the storage subsystem during the accounting period covered by the last successful accounting run. Set after a successful run.

Buttons

Start Accounting

Causes the accounting utility to start running for the storage subsystem. As it runs, the status fields will be updated after the statistics have been changed by the accounting utility.

25.2.2.3. Interpreting the accounting report

The accounting program for HPSS generates a JSON file that contains eight members.

"Date Reported"

The date when the report completed.

"Subsystem"

The subsystem for which the report was run.

"Number of Accounts"

Total number of accounts identified during the accounting run.

"Storage Unit Size"

The size of the units reported in the "Capacity Report", "Bandwidth Report", and "Bandwidth Summary".

"User Tuple"

The ordering used for the report. Each token in this string with be the key to each member in the "Capacity Report", "Bandwidth Report", and "Bandwidth Summary".

"Capacity Report"

An array of objects, each representing accounting information for a unique "User Tuple". The objects will each have the members listed in the "User Tuple" as well as:

"COS"

The class of service.

"Last Mod Time"

The last time this user read/wrote the COS.

"Files Stored"

Total number of files owned by this user in this COS.

"Units Stored"

Total number of units of said "Files Stored".

"Bandwidth Report"

An array of objects, each representing accounting information for a unique "User Tuple". The objects will each have the members listed in the "User Tuple" as well as:

"COS"

The class of service.

"SClass"

The storage class.

"Last Mod Time"

The last time this user read/wrote the COS/SClass.

"Number of Reads"

Number of individual read operations performed on this COS/SClass. In general, reads are counted against the user accessing the file. If a COS is configured to stage bitfiles on open, then all reads will occur in the SClass at the top of the hierarchy.

"Number of Writes"

Number of individual write operations performed on this COS/SClass. In general, writes are counted against the user accessing the file.

"Units Read"

Total number of units read from this COS/SClass.

"Units Written"

Total number of units written from this COS/SClass.

"Bandwidth Summary"

An array of objects, each representing accounting information for a unique "User Tuple". This summarizes the information in "Bandwidth Report" for each "User Tuple" and COS. Thus, this is identical to the "Bandwidth Report" minus the "SClass" field. With the exception of the "Last Mod Time" member, the members are summations of the information from the "Bandwidth Report" for each COS.

Note: File transfer information is not always accounted for when it occurs through interfaces other than the HPSS Client API.

Example accounting report file:

```
{
  "Date Reported": "Fri Mar  8 12:38:04 2019\n",
  "Subsystem": 1,
  "Number of Accounts": 1,
  "Storage Unit Size": 1,
  "User Tuple": "AccountId, UID, GID",
  "Capacity Report": [
    {
      "AccountId": 2,
      "UID": 3,
      "GID": 4,
      "COS": 5,
      "Last Mod Time": 12345,
      "Files Stored": 2,
      "Units Stored": 800
    }
  ],
  "Bandwidth Report": [
    {
      "AccountId": 2,
      "UID": 3,
      "GID": 4,
      "COS": 5,
      "SClass": 6,
      "Last Mod Time": 7054,
      "Number of Reads": 100,
      "Number of Writes": 53,
      "Units Read": 8024,
      "Units Written": 80000
    },
    {
      "AccountId": 2,
      "UID": 3,
      "GID": 4,
      "COS": 5,
      "SClass": 1,
      "Last Mod Time": 12345,
      "Number of Reads": 42,
```

```
      "Number of Writes": 91,
      "Units Read": 103,
      "Units Written": 623
    }
  ],
  "Bandwidth Summary": [
    {
      "AccountId": 2,
      "UID": 3,
      "GID": 4,
      "COS": 5,
      "Last Mod Time": 12345,
      "Number of Reads": 142,
      "Number of Writes": 144,
      "Units Read": 8127,
      "Units Written": 80623
    }
  ]
}
```

Sites may also wish to correlate the accounting report with a site-provided Account Map (see Accounting procedures) to determine the appropriate accounting charges.

The HPSS accounting report and a copy of any current site-defined accounting configuration files, if used, should be named with the date and time and stored for future reference. Site administrators may choose to write scripts to copy or archive the generated accounting report file from its original location.

25.2.3. Accounting procedures

The amount of work needed to support accounting varies according to the style of accounting used, whether or not account validation is being used, and whether the site requires additional information beyond that maintained by HPSS.

For UNIX-style accounting, with or without account validation, no additional work is needed to manage accounting.

For SITE-style accounting, the site must define and maintain a mapping of user IDs to account IDs. For each user, the mapping must define the user's default account ID and any other accounts for which the user is authorized.

- If Account Validation is enabled:
  - The user ID to account ID mapping is maintained in the Account Validation table. This table can store the default account ID of each user and any other accounts for which the user is authorized. The site must maintain this table. HPSS provides the table and a tool for maintaining it, the Account Validation Metadata editor (hpss_avaledit(7)).
  - If the site requires additional information beyond that supplied by the Account Validation table, it must create and maintain a site-defined Account Map. See Site-defined accounting configuration files and procedures.

- For account validation to work, a Gatekeeper must be configured. Also, the **Account Validation** flag must be enabled on the *Accounting Policy* window.

- If Account Validation is disabled:
  - The site will need to create a local site Account Map to maintain a list of the account IDs for which each user is authorized. This is a locally designed and written table, not supported by HPSS. See Site-defined accounting configuration files and procedures for suggestions for designing and maintaining an Account Map.
  - Administrators will need to manage user default accounts by updating the user's LDAP registry information (that is, the "AA=<default-acct-idx>" string in the user's hpssGECOS field).

25.2.3.1. Site-defined accounting configuration files and procedures

This section describes situations in which a site would need additional tables or procedures beyond those provided by HPSS. It offers suggestions on ways to implement these tables and procedures.

Site-defined Account Maps

If a site does not use account validation or if the site requires additional information beyond that maintained in the Account Validation table, it will need to create a local site-defined Account Map. For example, some sites may need to keep track of a department or company name. The Account Map is designed, created, and maintained by the site, not by HPSS.

The HPSS Account Index of each user should correspond to an Account Map entry that has the site information. The following are examples of possible Account Maps:

SITE-style Account Map:

```
Acct        User UID   Charge         Update_flag
12          dlk  3152  5A12x401       0
27          dlk  3152  5A12x501       0
341         dlk  3152  5A12x601       0
469         dpc  1478  7A14x401       0
470         dpc  1478  7A14x501       0
471         dmb  5674  5A12x401       0
7111        dmb  5674  7A14x501       0
...         ...  ...   ...            ...
```

UNIX-style Account Map:

```
UID       Charge
1478      7A14x401
3152      5A12x401
5674      5A12x401
...       ...
```

Note that if SITE-style accounting is in use and Account Validation has been enabled, the user-to-

account index mappings are already maintained in the Account Validation table. If no additional mappings are needed, there is no need for the site to create a separate site-defined Account Map.

Site-defined account apportionment table

In UNIX-style accounting, the UID (as Account Index) maps only to a specific user. Some sites may wish to apportion different percentages of the charges for a single UID among different project charge codes, but without using SITE-style accounting. HPSS does not provide a means to do this, but a site could implement its own table and utilities to do so. Here is an example of an account apportionment table which a site might implement to do this:

```
UID       % of (Project(s))
1478      75(DND) 25(CBC)
3152      45(DDI) 25(DND) 30(CBC)
5674      100(DDI)
...       ..................
```

The site would need to write utilities to tie the apportionment table to the HPSS accounting report.

Maintaining site-defined accounting files

Sites which require a site-defined Account Map, apportionment table, or other accounting files must develop tools to maintain and modify the files according to local accounting policies. Tools will be necessary to change, add, or delete an HPSS Account Index and its associated information. Site administrators must implement their own policies on what to do when an account is deleted, a user moves to another project, or a user leaves the system.

Other needed mappings, if any, must be maintained by the site in the Account Map file. When Account Validation is disabled, the site will need to maintain the user-to-account index mappings in the Account Map file as well.

UNIX-style accounting changes of this nature are handled through the normal utilities that set up and modify users and UIDs. A basic set of site-developed utilities are described in more detail below.

Add a user and account

New entries can be made and the next available HPSS Account Index number will be assigned from the free list. The free list will most likely consist of the last assigned number plus one, but could include reclaimed index numbers if a site chooses to reuse Account Index numbers that were previously assigned and no longer referenced. It is not likely that a site will need to reclaim Account Index numbers, but it is an option.

Delete a user

When a user is deleted from the Account Map, the HPSS files must be reassigned to another HPSS Account Index. This should be done from the HPSS client interface side. The **update_flag** should be set to **true** to indicate that this account index number can be reclaimed. The reclaiming tool should check for files using the account number before reclaiming it. When the Account Index is reclaimed, it can be put on the free list to be reused. It is important to keep a

copy of the Account Map that corresponds to the HPSS accounting snapshot of storage space in use for that time period so that the proper site information can be matched.

Delete account

When an account is deleted, it is handled in the same manner as when a user is deleted.

Modify account

The entries in the Account Map can be modified to correlate the Account Index to different information, but care should be taken to keep a copy of the corresponding tables for past HPSS accounting runs.

25.2.3.2. Accounting intervals and charges

The time between accounting runs and the charging policy for space usage should be developed after consulting the accounting requirements. The following are some guidelines to consider:

- Accounting should be run at a regular intervals, such as once per month.
- An accounting run may take several minutes, and the storage system will probably be active during the run. The resource usage reported for each user will reflect the resources used by that user at the point when the accounting run encounters that user. This is why accounting represents a blurred snapshot instead of a snapshot at a single point in time.
- Certain accounting information is kept in cache for several minutes after it has changed. For this reason, changes to a user's accounting data may not appear in an accounting report until this period of time has elapsed. Those changes which are still in cache when accounting runs will not appear on the current accounting report, but will appear on the next accounting report.
- The number of file accesses and the amount of data transferred can be taken to represent the activity level of a certain user account in the HPSS system. You may wish to charge specifically for the network and server resources consumed by this activity.
- It may be useful to charge different rates for each Class of Service. For example, a Class of Service that keeps two tape copies of each file will use up more tape cartridges than a Class of Service that keeps only a single copy of each file on tape.

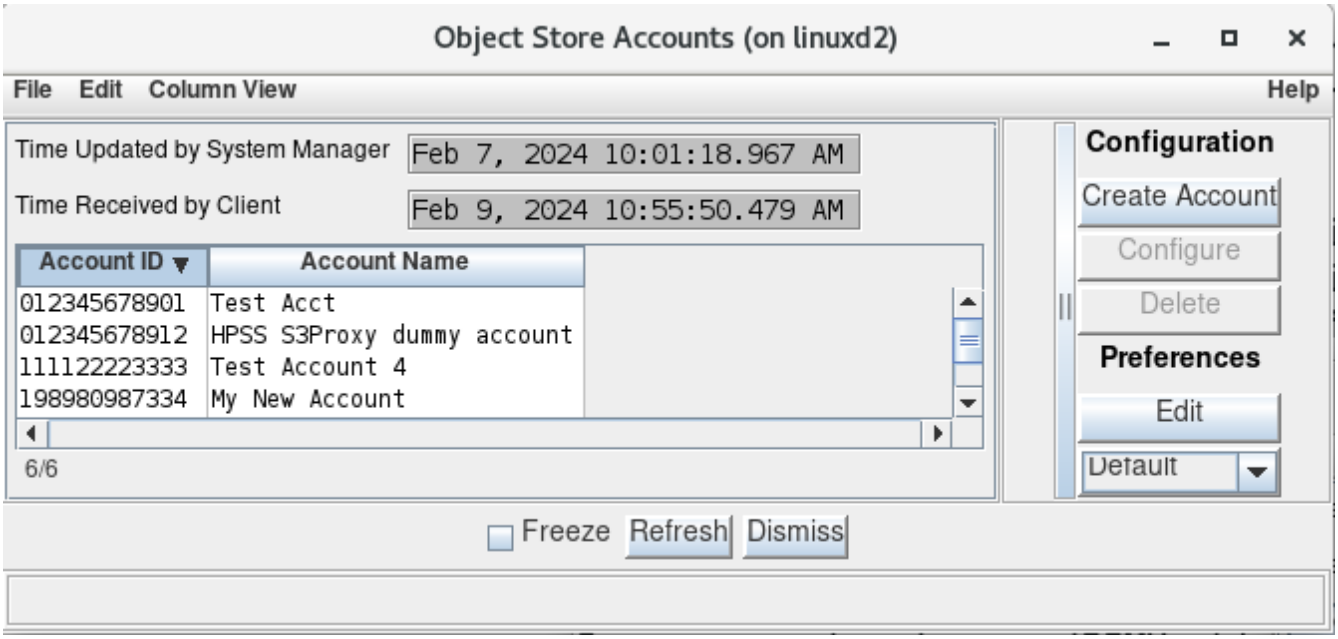
Chapter 26. Object store accounts and accounting

The object store account used when accessing an object store via HPSS is based on a configured object store mapping. These mappings associate HPSS accounts and storage classes to an object store bucket and identity. Configuration of these mappings is described in the following sections.

26.1. Managing object store accounts

26.1.1. Object Store Accounts window

An object store account can be created and managed using the *Object Store Accounts* window.



This window displays all the configured object store accounts. This window is displayed by first selecting **Configure** from the *HPSS Health and Status* window. This will display a drop-down menu. Select **Object Store Account** from this menu. This will display a drop-down menu. Select **Accounts** from this menu.

Field descriptions

Object Store Accounts List

The central panel in this window shows the list of configured object store accounts. The columns display selected fields from the object store account configuration windows discussed in the following sections.

Configuration buttons

Create Account

Opens a *Object Store Account Configuration* window containing default values for a new object

store account.

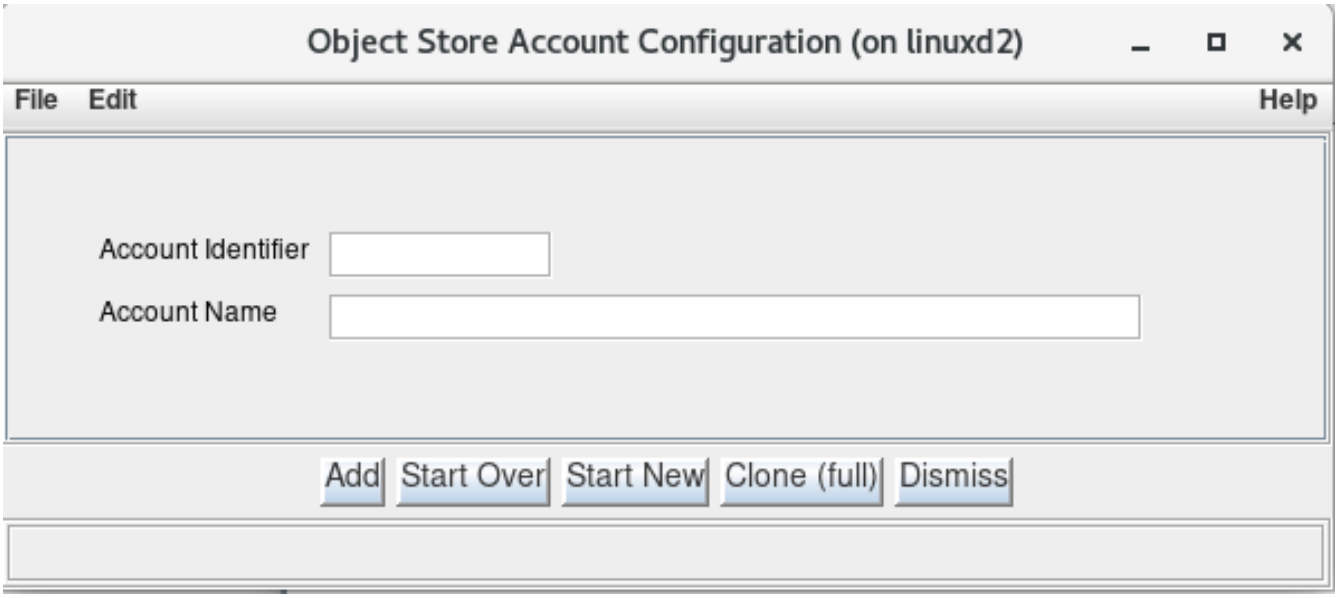
Configure

Opens the selected object store account configurations for editing.

Delete

Deletes the one or more selected object store accounts.

26.1.2. Object Store Account Configuration



This window is used to manage object store account configurations.

Field descriptions

Account ID

The 12 digit numeric identifier associated with an existing object store account.



Advice

This will be the AWS Account Identifier associated with your AWS account. This can be found via the AWS console. Login to the AWS console and select the account name button at the top right. The account number is the dash delimited digits displayed in the pull down. For example: 0123-4567-8901.

Account Name

The descriptive name of the object store account.



Advice

This a user-friendly name to help easily identify object store account configurations.

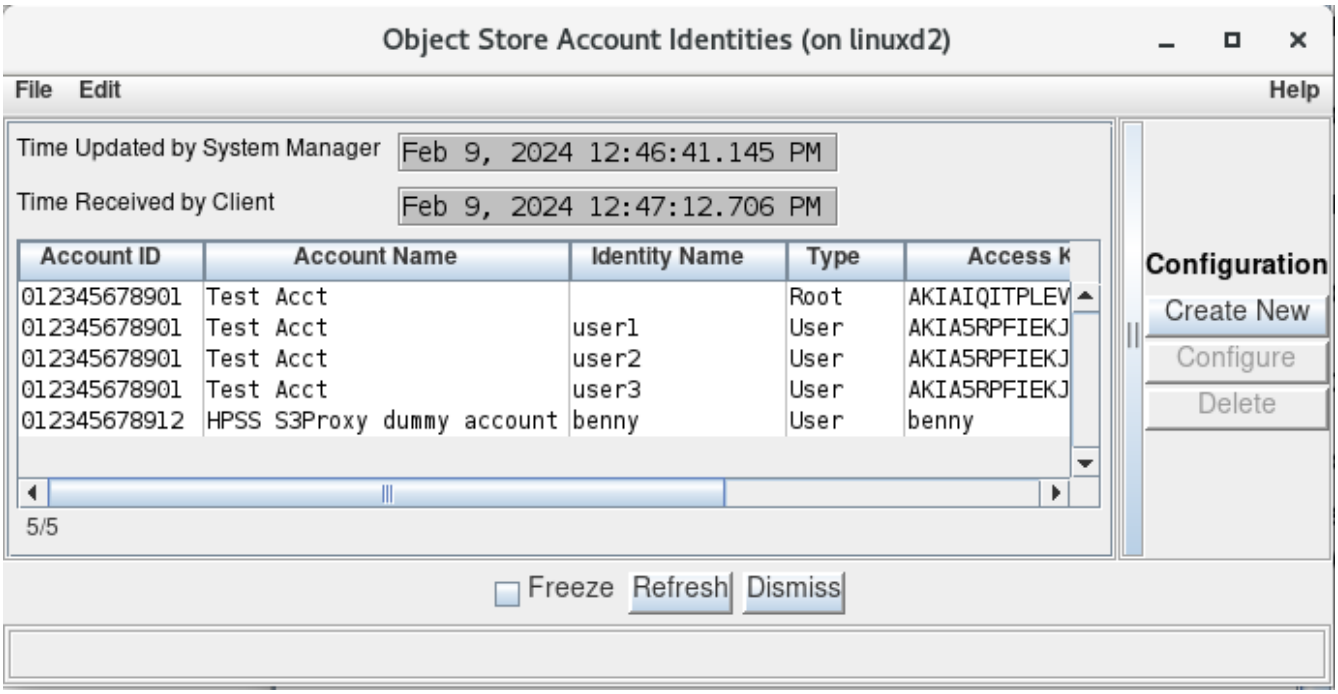
26.2. Managing object store account identities

Object store account identity configurations are used to authenticate HPSS when communicating

with the object store.

26.2.1. Object Store Account Identities window

An object store account identities can be created and managed using the *Object Store Account Identities* window.



This window displays all the configured object store account identities. This window is displayed by first selecting **Configure** from the *HPSS Health and Status* window. This will display a drop-down menu. Select **Object Store Account** from this menu. This will display a drop-down menu. Select **Identities** from this menu.

Field descriptions

Object Store Account Identities List

The central panel in this window shows the list of configured object store account identities. The columns display selected fields from the object store account identity configuration windows discussed in the following sections.

Configuration buttons

Create New

Opens a *Object Store Account Identity Configuration* window containing default values for a new object store account identity.

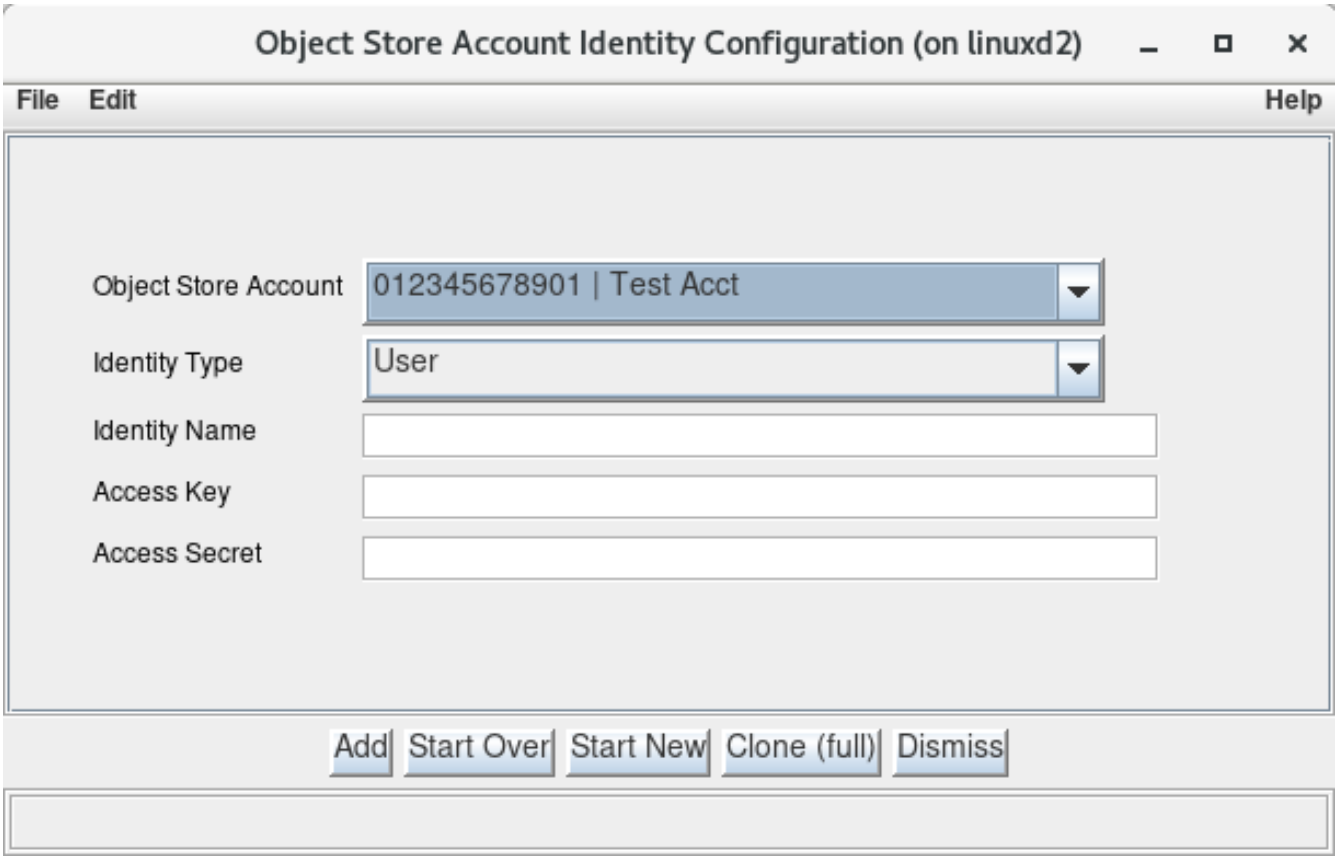
Configure

Opens the selected object store account identity configurations for editing.

Delete

Deletes the one or more selected object store account identities.

26.2.2. Object Store Account Identity Configuration



This window is used to manage object store account identity configurations.

Field descriptions

Object Store Account

The object store account associated with this identity.



Advice

This will be the AWS Account Identifier associated with your AWS account. This can be found via the AWS management console. Login to the AWS console and select the account name button at the top right. The account number is the dash delimited digits displayed in the pull down. For example: 0123-4567-8901.

Identity Type

The account identity type, either *root* or *user*.



Advice

*The **root** type is used when the identity represents the AWS root account. Using the root account is supported but discouraged of using IAM **user** accounts for identities.*

Identity Name

The name for this account identity.



Advice

This name corresponds to the AWS IAM identity that will be used when authenticating with AWS. You can locate this name via the AWS management console. If using an AWS root account, no name is provided.

Access Key

The access key used to authenticate this identity.



Advice

This is the 20 character alphanumeric access key associated with the AWS identity (root or IAM). This value can be generated via the AWS management console.

Access Secret

The access secret used to authenticate this identify.



Advice

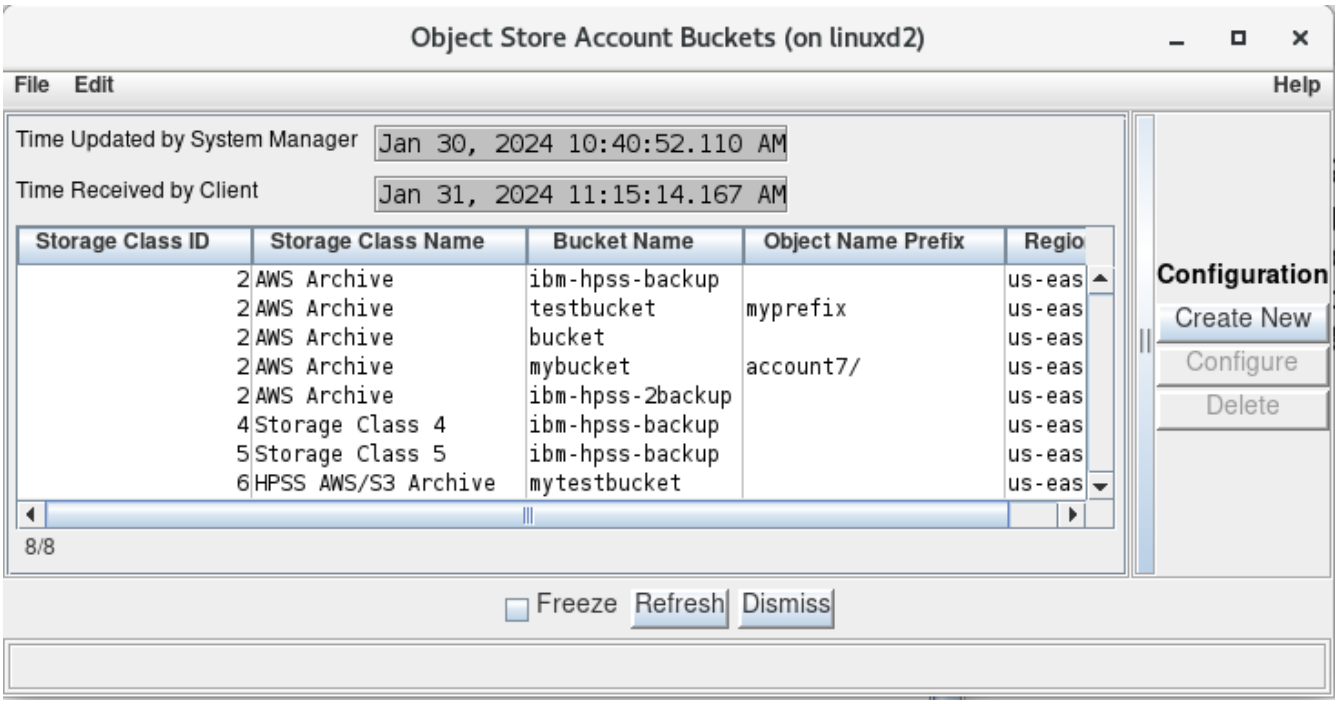
This is the access secret that can be downloaded from the AWS management console when a new access key is generated for an account.

26.3. Managing Object Store Account Buckets

The account bucket configuration associates object store buckets, regions, and optional key prefixes with an HPSS storage class.

26.3.1. Object Store Account Buckets window

An object store account bucket can be created and managed using the *Object Store Account Buckets* window.



This window displays all the configured object store account buckets. This window is displayed by

first selecting **Configure** from the *HPSS Health and Status* window. This will display a drop-down menu. Select **Object Store Account** from this menu. This will display a drop-down menu. Select **Buckets** from this menu.

Field descriptions

Object Store Account Buckets List

The central panel in this window shows the list of configured object store account buckets. The columns display selected fields from the object store account bucket configuration windows discussed in the following sections.

Configuration buttons

Create New

Opens a *Object Store Account Bucket Configuration* window containing default values for a new object store account bucket.

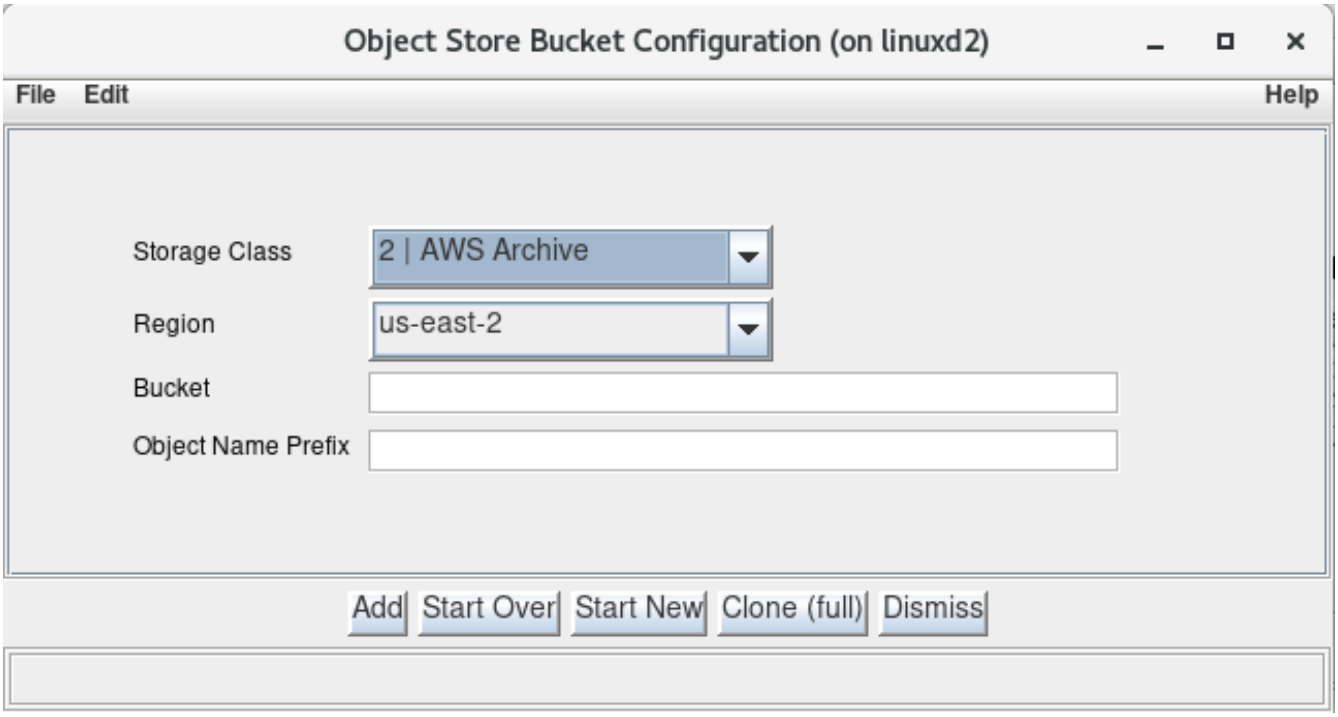
Configure

Opens the selected object store account bucket configurations for editing.

Delete

Deletes the one or more selected object store account buckets.

26.3.2. Object Store Account Bucket Configuration



This window is used to manage object store account bucket configurations.

Field descriptions

Storage Class

The storage class with which to associate the bucket, region, and object key prefix combination.

Region

The region for the specified bucket.

Bucket

The name of the object store bucket.



Advice

This is the name of the object store bucket used to contain objects created during data migration.

Object Name Prefix

An optional path prefix to use when storing objects associated with this account bucket.



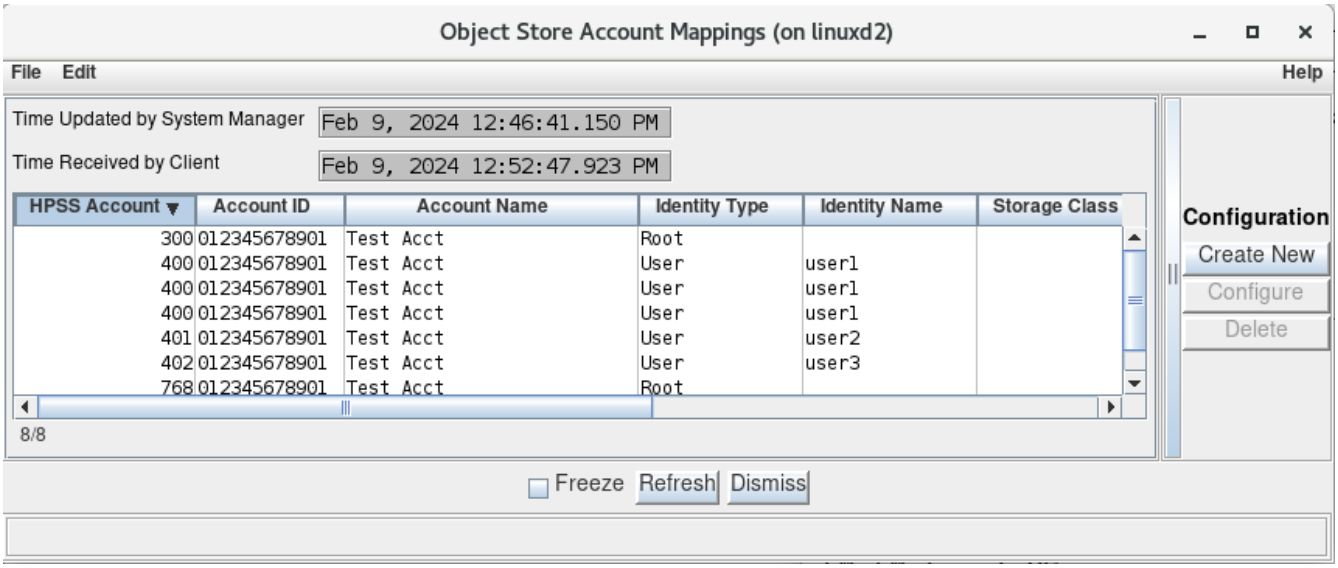
Advice

This can be used to logically partition the contents of the bucket.

26.4. Managing Object Store Account Mappings

26.4.1. Object Store Account Mappings window

An object store account mapping can be created and managed using the *Object Store Account Mappings* window.



This window displays all the configured object store account mappings. This window is displayed by first selecting **Configure** from the *HPSS Health and Status* window. This will display a drop-down menu. Select **Object Store Account** from this menu. This will display a drop-down menu. Select **Mappings** from this menu.

Field descriptions

Object Store Account Mappings List

The central panel in this window shows the list of configured object store account mappings. The columns display selected fields from the object store account mapping configuration windows discussed in the following sections.

Configuration buttons

Create Account

Opens a *Object Store Account Mapping Configuration* window containing default values for a new object store account mapping.

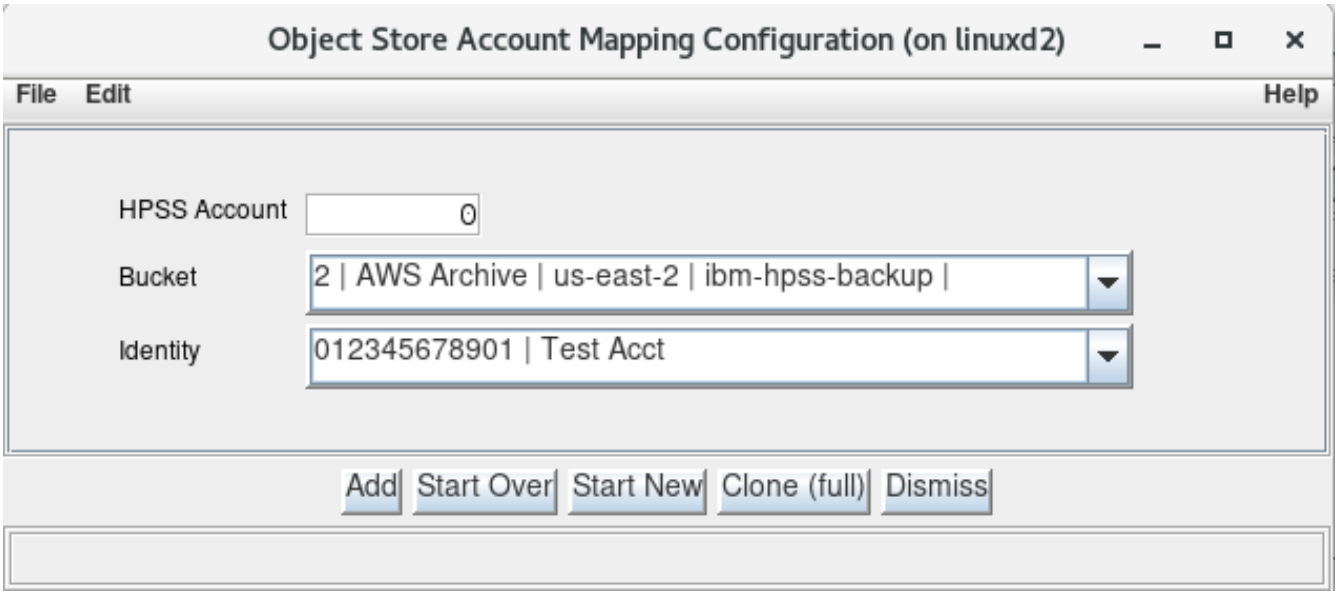
Configure

Opens the selected object store account mapping configurations for editing.

Delete

Deletes the one or more selected object store account mappings.

26.4.2. Object Store Account Mapping Configuration



This window is used to manage object store account mapping configurations.

Field descriptions

HPSS Account

The HPSS numeric account identifier mapped to the specified account bucket and account identity combination.



Advice

See the User accounts and accounting for information about setting up and obtaining HPSS account identifiers.

Bucket

The account bucket used for the mapping.

Identity

The account identity used for the mapping.

Chapter 27. User interfaces

This chapter describes the configuration information for the user interfaces provided with HPSS for transferring files:

- Client Application Programming Interface (API)
- HPSS UNIX Tools (UNIX Tools)
- Parallel File Transfer Protocol (FTP) or PFTP

For other application configuration information, such as HSI or hpssfs(7), consult documentation bundled with the application.

27.1. Client API configuration

The following environment variables can be used to define the Client API configuration. The defaults for these variables are defined in the `hpss_env_defs.h` file.

HPSS_API_AUTHN_MECH

Specifies which authentication mechanism may be used. The values may be `krb5` or `unix`. The default is HPSS_CLIENT_AUTHN_MECH.

HPSS_API_BLOCK_SIZE

Specifies the preferred block size for efficient file system I/O. The default is 1048576 bytes (1MiB).

HPSS_API_BUSY_DELAY

Specifies the number of seconds to delay between retry attempts. Note that this value is used both for retrying initialization operations (see HPSS_API_RETRIES) and Core Server requests (See HPSS_API_BUSY_RETRIES). The default value is "15".

HPSS_API_BUSY_RETRIES

Specifies the number of retries to be performed when a request fails because the Core Server does not currently have an available thread to handle that request. A value of zero indicates that no retries are to be performed (that is, the operation will be performed once), and a value of "-1" indicates that retries should be attempted until either the request succeeds or fails for another reason. The default value is "3" (which will result in up to four attempts).

HPSS_API_DEBUG

Specifies the level of debugging the Client API should produce. By default, no debug logging is produced. For more information on the debugging levels, consult the *HPSS Client API Programmer's Reference*. When API debugging is activated messages will go to stdout unless the application sets the log path or the HPSS_API_DEBUG_PATH environment variable is set. API debugging can be useful in tracing API problems or understanding network behavior of an HPSS application.

HPSS_API_DEBUG_PATH

Specifies the destination of debug messages; default is `stdout`. If HPSS_API_DEBUG_PATH is set to `stdout` or `stderr`, debug messages will be written to the standard output or standard error I/O streams, respectively. Otherwise, the value will be interpreted as the pathname of a file. The Client API must be compiled with debugging enabled.

HPSS_API_DISABLE_CROSS_REALM

Specifies cross-realm authentication. When cross-realm authentication is disabled, a client will not be allowed to access directories which are located in another security realm. The default value is zero (meaning that cross-cell authentication is enabled). To disable cross realm authentication, HPSS_API_DISABLE_CROSS_REALM must be set to a numeric, nonzero value.

HPSS_API_DISABLE_JUNCTIONS

Specifies whether junction traversal is enabled. When junction traversal is disabled, a client will not be allowed to access directories which require traversal to the directory via a junction. The default value is zero (meaning junction traversal is enabled). To disable junction traversal, HPSS_API_DISABLE_JUNCTIONS must be set to a numeric, nonzero value.

HPSS_API_DMAP_WRITE_UPDATES

Specifies the frequency of cache invalidates that are issued to the DMAPI file system while writing to a file that is mirror in HPSS. The value indicates the number of write operations between cache invalidates. The default value is "20".

HPSS_API_HOSTNAME

Specifies the hostname to be used for TCP/IP listen ports created by the Client API. The default value is HPSS_HOST. This value can have a significant impact on data transfer performance for data transfers that are handled by the Client API (that is, those that use the **hpss_Read** and **hpss_Write** interfaces). The default is HPSS_HOST.

HPSS_API_LIMITED_RETRIES

Specifies the number of retry attempts before a limited retry error operation fails. Currently, no operations are defined as "limited retry error operations". The default value is "1".

HPSS_API_MAX_CONN

Specifies the number of connections that are supported by the Client API within a single client process. If HPSS_API_MAX_CONN is set to zero, the number of connections is equal to the default supported by the HPSS connection management software - currently 20. If HPSS_API_MAX_CONN is nonzero, it is the number of connections to be used.

HPSS_API_RETRIES

Specifies the number of retries to attempt when an operation fails. Currently this class of operation includes library initialization and communications failures. A value of zero indicates that no retries are to be performed (that is, the operation will be attempted once), and value of "-1" indicates that the operation will be retried until successful. The default value is "4" (meaning the operation will be attempted up to five times).

HPSS_API_RETRY_STAGE_INP

Specifies whether retries are attempted on attempts to open files in a Class of Service that is

configured for background staging on open. A numeric, nonzero value (the default) indicates that opens which would return **-EINPROGRESS** to indicate that the file is being staged will be retried (using HPSS_API_RETRIES and HPSS_API_BUSY_DELAY to control the number and timing of retries), while a value of zero indicates that the **-EINPROGRESS** return code will be returned to the client. The default value is "1".

HPSS_API_REUSE_CONNECTIONS

Specifies whether TCP/IP connections are to be left open as long as a file is open or are to be closed after each read or write request. A numeric, nonzero value will cause connections to remain open, while a value of zero will cause connections to be closed after each file access. The default value is zero.

HPSS_API_RPC_PROT_LEVEL

Specifies the RPC protection level. Valid values are `connect`, `packet`, `packet integrity`, and `packet privacy`. The default is HPSS_RPC_PROT_LEVEL.

HPSS_API_SAN3P

Specifies whether to enable SAN3P transfers. Valid values are "on" and "off". The default is on.

HPSS_API_TOTAL_DELAY

Specifies the number of seconds to continue retrying requests. A value of zero indicates no time limit. The default value is zero.

HPSS_API_USE_PORT_RANGE

Specifies whether the HPSS Movers should use the configured port range when making TCP/IP connections for read and write requests. A numeric, nonzero value will cause Movers to use the port range, while a value of zero will cause Movers to allow the operating system to select the port number. The default value is zero.

HPSS_API_PORT_RANGE

Specifies the port range that the HPSS Client API will use when binding to a port. The client API normally binds to any available port. If the HPSS_API_PORT_RANGE is set, then the Client API will attempt to bind to one of the ports in the specified range. The environment variable can be set in the env.conf file or directly in the environment. The range is specified as HPSS_API_PORT_RANGE=X-Y where X and Y are positive integers >= 1024 and ⇐ 65535. For Client API applications that make use of hpss_net_bind(), please update your application to use hpss_NetBind() to ensure that the application respects the HPSS_API_PORT_RANGE variable. This variable is unrelated to HPSS_API_USE_PORT_RANGE. The default value is NULL.

HPSS_KRB5_KEYTAB_FILE

Specifies the name of the file containing the security keys necessary for successfully initializing the Client API for Kerberos authentication. The default is `auth_keytab:/var/hpss/etc/hpss.keytab`.

HPSS_UNIX_KEYTAB_FILE

Specifies the name of the file containing the security keys necessary for successfully initializing the Client API for UNIX authentication. The default is `auth_keytab:/var/hpss/etc/hpss.unix.keytab`.

HPSS_SEC_MUTUAL_AUTH

Specifies whether clients should utilize mutual authentication security. This forces the client to be authenticated by the server (which always happens), and the server's credentials authenticated by the client. This can prevent certain man-in-the-middle attacks. The default is `true` (on). Due to current security subsystem limitations (such as in UNIX system authentication), mutual authentication may be turned off by setting this variable to `off`. There is some amount of overhead to mutual authentication for first-time client startup, which may be another reason a site would disable it.

HPSS_API_CIPHER

Specifies the cipher to use for data transfers. Run `hpss_list_enc enc` for a list of ciphers. Note that configuration policy can restrict which ciphers are allowed.

HPSS_API_ENCRYPT

Set to a nonzero value to force encryption on data transfers.

HPSS_API_KEY_LEN

Set to the size of encryption key used by the API. This is not necessarily the same as the key size for the cipher. The key is hashed through a key expansion algorithm to produce enough bytes for the cipher. This allows the API key size to be larger than the cipher size, and possibly even smaller (not recommended). Default is 32 bytes (256 bits).

27.2. HPSS UNIX Tools Configuration

The HPSS UNIX Tools respond to the HPSS API configuration items mentioned above. They do not have a shared configuration file.

User authentication uses the current system user, or the **HPSS_CLI_USER** environment variable if it is set to specify the HPSS principal. The tools also support typical authentication by command-line options, with that overriding the behavior defined by the environment.

A common setup for a UNIX environment would include a user creating a keytab with their HPSS user credentials in `~/keytab`, and setting up their user environment to have **HPSS_CLI_USER** set to their HPSS user. For kerberos, the user will by default use their existing kerberos credentials with no additional setup.

27.3. FTP/PFTP daemon configuration

The pftp_client binary is an enhanced FTP client providing parallel data transfer facilities, HPSS-specific features, and transfer of files greater than 4 GB. The HPSS binary, hpss_pftpd(7), is an enhanced FTP Server facilitating parallel transfers into HPSS. This binary *only* functions with HPSS. If you are using an "ftp" binary or a pftp_client(7) binary to access HPSS the site *must* run the hpss_pftpd binary. The hpss_pftpd(7) binary provides FTP services for both the standard system FTP client and the parallel-enhanced HPSS pftp_client.

Note: Not every feature of every system FTP client will be honored by the hpss_pftpd. Additionally, system FTP clients cannot take advantage of the parallel transfer features nor the increased file size capacity provided by the hpss_pftpd. If a site wishes to provide FTP services to both the system FTP

server, ftpd, and HPSS, hpss_pftpd, it will be necessary to set up the two services to listen on different ports.

There are four steps that must be performed prior to using the FTP user interface to transfer HPSS data:

1. Verifying the FTP initial configuration
2. Configuring the FTP daemon syslog
3. Defining the FTP access
4. Creating FTP users

These steps are described in more detail in the paragraphs that follow.

Step 1. Verifying the FTP configuration

Configuring FTP may differ depending on whether it is done via xinetd or systemd.

xinetd

During the HPSS infrastructure configuration phase, the following files were created or modified by the **mkhpss** script:

- In the `/etc/services` file, verify that the hpssftp and hpssftp-data entries were added correctly. Ports 20 and 21 are the ports of choice to prevent having to educate every user to alternate ports. By default, **mkhpss** uses ports 4020/4021. It may be necessary to move the existing ftp and ftp-data to other ports when assigning the hpssftp and hpssftp-data to ports 20/21.

systemd

- Review the `hpssftp.socket.systemd.tmpl` and `hpssftp@.service.systemd.tmpl` files for correct executable paths, desired port number, and other desired systemd options.
- Use the following steps to install and enable the HPSS FTP service.

```
cp /opt/hpss/config/templates/system_files/hpssftp.socket.systemd.tmpl \
/etc/systemd/system/hpssftp.socket
cp /opt/hpss/config/templates/system_files/hpssftp@.service.systemd.tmpl \
/etc/systemd/system/hpssftp@.service
systemctl enable hpssftp.socket
systemctl start hpssftp.socket
systemctl status hpssftp.socket
```

The systemd status should be enabled and running.

General

After verifying the PFTP service, verify the remaining configuration items.

A "TCP Wrapper" application may be used for initiating the HPSS PFTP Daemon to make it more secure; however, such a program is not provided by the HPSS project.

The file /var/hpss/etc/HPSS.conf provides the options for both an HPSS PFTP daemon and a non-HPSS PFTP daemon (DISCOM PFTP daemon). This file should be customized as needed. Refer to the HPSS.conf(7) section or the HPSS.conf configuration file for details.



It may be necessary for a site to merge older copies of the HPSS.conf file with the template if modifications have been made since the prior release. There is no conversion job to perform this task.

Step 2. Configuring the FTP/PFTP daemon syslog

The PFTP daemon attempts to write to the system log (using **syslog()**). To enable this output, follow local system procedures for capturing this information. The PFTP daemon writes to the **LOG_DAEMON** syslog facility.

Step 3. Defining the PFTP access

In the following statements, **{FTPBaseDir}** defaults to "/var/hpss" and **{ftpaccess}** defaults to "ftpaccess". These values may be changed in the HPSS.conf file. The **{FTPBaseDir}** /etc/ **{ftpaccess}** file defines access, options, and restrictions for the server. An ftpaccess.tmpl file is provided in the $HPSS_ROOT/config/templates directory for the HPSS administrator to customize appropriately. The pound sign (#) at the beginning of a line indicates a comment. Options are enabled by removing the comment symbol (#) and conversely disabled by introducing this symbol. Some options are specific to individual machines, so each HPSS PFTP daemon should read its own **{ftpaccess}** file.

The access options can be defined as follows:

```
# Define a class of users.
# class <class> { anonymous | guest | real } <user> [ <user> ... ]
#
# <user> is a full network address, including wildcarding, e.g.,
*.nsl.nersc.gov.
class hpss_class real *
class hpss_anon anonymous *
#
# Define a limit on the number of users from a certain class
# that can be using the ftp server at specific times.
# limit <class> <n> [<times>] [<msg_file>]
#
# The <times> clause has the following format:
# <day><time>-<time>|<day><time>-<time>|... (no white space!)
# or "Any"
# where <day> is one of "Su", "Mo", "Tu", "We", "Th", "Fr",
# "Sa", "Wk"
# "Wk" matches any day of the week
# <time> is the time in 24-hour format
# "Any" matches any day and time
# (used to specify <msg_file> for all days and times)
# limit hpss_class 1 Tu0600-1200
limit hpss_class 3
#
```

```
# Display a banner prior to the user prompt (before log in.) Very
# useful for informing the user that he/she is dealing with the
# HPSS file systems rather than the local file system of the
# host executing the FTP Daemon. This is highly recommended!

banner <pathname/filename>

# Control for logging (sent to syslog()).
log [ commands ] [ anonymous ] [ { inbound } ]
[ transfers ] [ guest ] [ { outbound } ] [ real ] [debug]

# Set the maximum number of login attempts before closing the
# connection:
loginfails <login_failure_threshold>

# Determine the appropriate behavior when needed data must be staged.
# <stage-code> can have several values:
# -1 wait forever
# 0 do not wait (default) - inform user that data needs staging and
# exit
# >0 wait for n seconds - if staging takes more than n seconds, inform
# the user and exit
# note: This option requires that the relevant storage class be
# configured in a compatible way, or behavior may not be as
# expected. For example, setting wait_on_stage to -1 (wait forever)
# for a SC with no-stage set would make little sense.
wait_on_stage <stage-code>
#
# Define users as being guests (which means they cannot
# cd out of their own subdirectory).
guestgroup <group> [ <group> ... ]

# Deny access to a set of users, <addrglob> is full network address,
# including wildcarding, e.g., *.nsl.nersc.gov.
deny <addrglob> [ <msg_file> ]

# Send shutdown message to current FTP users.
shutdown <shutdown_info_pathname>

# Send message to users, possible on access to a directory.
message <pathname> [ <when> ]

# Display prompt to read a file, possible on access to a directory.
readme <pathname> [ <when> ]
# The <when> clauses may be either "login" (in which case the file
# is displayed when the user logs in) - Applicable only for anonymous
# or "cwd=[<directory_name> | *]",
# (in which case the file is displayed when the user changes to the
# named directory, with '*' wildcarding all directories).
# ONLY displays message on FIRST entrance. Directory MUST be full path
```

- **Message/readme/banner/shutdown** (message lines) are text files, with the following keywords (all one character in length) recognized, identified by a preceding **%**:

  Table 32. Banner keywords

Keyword	Description
M	Class limit
T	Current time on FTP Daemon
C	Current working directory
R	Remote hostname
L	Local hostname
U	User name
s	Shutdown time
d	User disconnect time
r	Connection deny time
%	%

- The format of the <**shutdown_info_pathname**> file is:

```
<year> <mon> <day> <hour> <min> <deny> <disc>
```

  Message lines contain keywords mentioned above. For example:

```
1994 1 28 16 0 120 30
System shutdown at %s (current time is %T)
New FTP sessions denied at %r
FTP users disconnected at %d
```

  indicates that shutdown will be 1/28/94 at 16:00, with users disconnected at 15:30 and sessions denied at 14:40 (the 120 indicates 1 hour, 20 minutes).

- The <**when**> clauses may be either **login** (in which case the file is displayed when the user logs in) or **cwd=[**<**directory_name**> | ***]** (in which case the file is displayed when the user changes to the named directory, with an asterisk (*) wildcarding all directories).
- The <**times**> clause has the following format:

  <**day**><**times**>-<**time**> | <**day**><**time**>-<**time**> | . . .

  where <**day**> is one of **Su, Mo, Tu, We, Th, Fr, Sa, Su, Wk**, or **Any** and <**time**> is the time of day on a 24-hour clock. For example, to specify Tuesday between 8:00 AM and 4:00 PM:

  Tu0800-1600

  The **Wk** entry for the <**day**> clause indicates a weekday (Monday through Friday), and **Any**

indicates all days of the week.

Step 4. Creating FTP users

In order for an HPSS user to use FTP, a UNIX or Kerberos (or both) userid and password must be created. Refer to The **hpssuser** utility for information on how to use the hpssuser(7) utility to create the userid and password and set up the necessary configuration for the user to use FTP. Note that this step should not be done until the Core Server is running so that the hpssuser(7) utility can create the home directory for the FTP user.

The **/opt/hpss/bin/hpss_pftppw** utility can be used to set the encrypted passwords in the `/var/hpss/etc/passwd` file. The syntax for this utility is as follows:

```
hpss_pftppw <userid> [<password file pathname>]
```

The utility will prompt the user for the old and new passwords. If root has to change the password it is necessary to manually edit the file and remove the existing encrypted password. The password file pathname argument can be used to specify a password file other than the default file, `/var/hpss/etc/passwd`.

To enable anonymous FTP, the "**hpssftp**" user must be defined in either the HPSS FTP password file or in the Kerberos KDC and LDAP registry (depending on which authentication mechanism is enabled). In addition, the entry for the "**hpssftp**" user must contain a home directory defined to be a non-NULL value. Anonymous FTP users will have the "root" directory set to "`/anonymous`" to prevent anonymous users from accessing other portions of HPSS. This means that for anonymous access the directory `/anonymous` *must* exist.

To disable anonymous FTP, do either or both of the following:

- Remove the hpss_anon class from the ftpaccess file
- Add **hpss_ftp**, **anonymous** or **guest** to the HPSS FTP user file (normally "`/var/hpss/etc/ftpusers`").

Security policies and the security requirements should be carefully examined and thoroughly understood before allowing "anonymous" FTP.

Chapter 28. Backup and recovery

This chapter discusses the following operations:

- Backup and recover HPSS metadata
- Backup HPSS environment
- Recover HPSS user data
- Handling DB2 space shortage

28.1. HPSS metadata backup and recovery

Each HPSS site is responsible for implementing and executing the HPSS metadata backup and recovery process. The HPSS administrator must ensure that HPSS metadata is backed up on a regular basis. In addition, the HPSS administrator must verify that the backup files can be used to restore the HPSS metadata up to the point of failure. Sites can choose to use a commercially available backup software tool, such as the Tivoli Storage Manager (TSM), to backup the HPSS metadata. Optionally, a set of simple and customizable backup scripts can be developed to backup the HPSS metadata. Contact HPSS support to obtain sample backup scripts and guidance for implementing the DB2 backup process for your site.

The following sections provide an overview of the metadata backup and recovery process.



The backup topics provided in the following sections are meant only as an overview. The DB2 Data Recovery and High Availability Guide and Reference should be consulted in order to obtain a complete understanding and explanation of issues and procedures associated with enabling full recoverability of DB2 data.

28.1.1. HPSS administrator responsibilities for DB2

While DB2 has sophisticated backup and recovery abilities, it is fundamentally the responsibility of the HPSS administrator to ensure that these facilities are configured and working correctly. This section outlines some of those responsibilities.

While storing the DB2 data tables on reliable disk storage systems can be very useful if it becomes necessary to recover the database, it is essential that the database be backed up regularly, and that the backup media be reliable. Many sites keep multiple copies of the backup media in separate locations. It is also extremely important that site administrators verify that they can restore a failed database from the backup media. Time should be scheduled on an occasional basis to perform practice recoveries from the backup media. This could be done by creating a separate instance of the HPSS DB2 system on a separate computer and restoring it from the primary instance backup media.

HPSS databases should be backed up regularly. Most sites back up the entire database once each night, using the online backup procedure. Note that Online backups are only available for database configurations that enable log archiving. (See Configuring DB2 for online backup.)

Online backups do not interfere with normal HPSS processing. It is equally important to monitor the online backup process and verify it is completing normally.

As HPSS operations take place, DB2 writes transaction log entries describing the changes that have been made to the database. These logs allow DB2 to recover all changes made to the database since the time of the last database backup, forward to the time when DB2 was stopped by a crash, hardware failure, power outage or whatever. It is vital that the DB2 log files be hosted on a highly reliable disk system. Furthermore, DB2 log mirroring must be used to protect this information on two separate disk systems. The disk systems must also be protected using RAID 1 or RAID 5 configurations. Reliable access to the log files is more important than performance. Loss of the DB2 log files can result in a significant impact to the operation of the system, extensive downtime to repair, and potential loss of end-user data. Proper protection of the DB2 log files and associated archived backups is critical to a site's HPSS operations.

Similarly, the HPSS administrator must regularly verify that completed DB2 log files are being stored on reliable media until they are no longer needed. Many sites make multiple copies of DB2 log files and store them in separate physical locations.

In case of a catastrophic failure of the HPSS database host computer, the database can be restored to a consistent state at the point in time of the failure, if the administrator has a reliable series of backup images and log files. After the failure that caused the database failure is corrected, the HPSS databases can be restored using the backup files, and then by replaying the DB2 log files. Much of this process happens automatically when DB2 is restarted, but if you have any questions about these requirements and procedures, contact HPSS support.

Another responsibility of the HPSS administrator is to periodically perform RUNSTATS operations on the database tables. RUNSTATS is a DB2 process that analyzes the tables and produces statistics that are vital to DB2's performance. Without good statistics, DB2 may create very inefficient database access plans that result in very poor performance. Although frequent RUNSTATS are not necessary on HPSS databases, the administrator should have a plan to perform them on some sort of regular schedule consistent with local requirements. You should discuss this with HPSS support.

It is also important to have a plan to perform REORGCHK operations from time to time to monitor the organization of the database tables. As a consequence of normal use, a database table may become significantly disorganized. Database performance can be enhanced by periodic reorganizations, some of which can be performed with HPSS in operation. You should also discuss this with HPSS support.

28.1.2. Overview of the DB2 backup process

Without proper and systematic metadata backup, a database failure could result in total loss of the HPSS system. For this reason, backing up the DB2 databases containing HPSS metadata is essential.

DB2 databases can be backed up in one of two ways:

1. Online backups are made that can be restored to the point of failure by "rolling forward" archived logs of transaction activity since the backup.

   This is the recommended method for making HPSS backups.

2. An offline snapshot of the database data is taken at a particular point in time.

   Offline backups provide recovery only to the point in time at which they were made, and they require that there be no other activity in the database while they are being made. *Therefore, this method is not recommended for use with HPSS.*

Since HPSS installations require the use of online backups, it is useful to understand the types of backup and restore operations that can be performed with this backup type. Online backups can be described as "full", "incremental", or "delta". Full backups record the full contents of the database at a point in time. Incremental backups record all the changed data since the last full backup. Delta backups record only the data that has changed since the most recent backup (of any type). Obviously, a full backup is necessary as a starting point in the recovery of a database. Incremental and delta backups help reduce the number of transaction logs that must be retrieved and applied during a restore operation.

Consider a site that keeps the following schedule for backing up a database:

```
Sunday: full
Monday: delta
Tuesday: delta
Wednesday: incremental
Thursday: delta
Friday: delta
Saturday: delta
```

Under this scheme, if the database fails between the Thursday and Friday backups, only a few backup images would be needed for a restore: Sunday's full backup is needed as a starting point. Wednesday's incremental backup includes all changes since Sunday, so it makes retrieval of Monday's and Tuesday's backups unnecessary. Beyond that, Thursday's delta backup is needed, as well as any transaction logs archived after Thursday's backup. So the needed images are: Sunday, Wednesday, Thursday, and only those logs between Thursday's backup and the point of failure.

The way DB2's backup and restore commands are implemented makes restoration simple. Basically, only the latest backup image needs to be specified. DB2 records the backup history in each backup, so it can determine which images and logs are necessary. Assuming an automated method is used for backup storage and retrieval, it can also fetch the images and restore from them. See the *DB2 Data Recovery Guide* for details.

28.1.2.1. Configuring DB2 for online backup

Several steps must be taken to prepare a DB2 database for online backup. Initially, a newly-created database is configured to perform offline backups. The database's configuration must be updated to enable log archiving. Online backups are only available when log archiving is enabled. An initial, offline backup must then be made as a starting point for the backup process. The database will be inoperative until this initial full backup is completed. An outline of the major steps to set up a database for online backup is:

1. Configure backup software and make certain data can be archived into it and retrieved from it.

2. Update the database configuration to enable log archiving, this will place the database in "backup pending" state.
3. Perform initial full, offline backup (database returns to normal state).
4. Schedule subsequent backups using **cron**.

To back up a DB2 database, invoke the **db2 backup** command using the DB2 command line processor. Typically, the administrator would put this command into a script along with any necessary parameters, such as the type of backup, the database partitions, the destination, and other important information required by DB2 to conduct its backup. The script would then be invoked from **cron**.

For details on configuring DB2 for backup, administrators should consult the *DB2 Data Recovery Guide* as well as the documentation for their backup software.

28.1.3. Overview of the DB2 recovery process

Upon discovering a damaged DB2 container, you must first determine the level of hardware recovery and problem determination available to you. For example, whether or not you were utilizing some level of RAID and can recover a failed disk from a good disk. The second step in recovering damaged or inaccessible DB2 containers is considering any software recovery tools available. For example, if you were utilizing backup software, such as TSM, this would enable you to replace the bad hardware and recover onto the new hardware device to a particular point in time. For details on recovering a DB2 database, see the IBM redbook titled "Data Recovery and High Availability Guide and Reference" for the appropriate version of DB2 installed on your system. This guide will help you decide how best to recover your DB2 database or damaged container.

28.2. HPSS system environmental backup

The HPSS administrator must also ensure that the HPSS system environment is backed up on a regular basis and to verify that the backups can be used to restore the environment. The following sections describe the environmental data that must be safeguarded.

28.2.1. HPSS file system backup

The following HPSS file systems should be backed up using the site backup procedure:

- `/opt/hpss`
- `/var/hpss`
- `/var/hpss/adm/core`
- `{HPSS metadata backup path}/cfg`
- `{HPSS metadata backup path}/subsys1`
- `/db2/backups/ldap` (if LDAP is used)
- `/var/hpss/hpssdb`
- `/var/hpss/hpssdb/ldap` (if LDAP is used)

- {DB2 log path}/cfg
- {DB2 log path}/subsys1
- {DB2 log path}/ldap (if LDAP is used)
- {DB2 log archive path}/cfg
- {DB2 log archive path}/subsys1
- {DB2 mirror log path}/cfg
- {DB2 mirror log path}/subsys1
- {DB2 mirror log archive path}/cfg
- {DB2 mirror log archive path}/subsys1
- {HPSS secondary metadata backup path}/cfg
- {HPSS secondary metadata backup path}/subsys1
- /<metadata backup file system>
- Other site-specific file systems

28.2.2. License file backup

While it should be accomplished by backing up /var/hpss as described in the section above, having backups of the license files provided by HPSS Support is important in the case that a license file has become corrupted, is accidentally deleted, or otherwise becomes unreadable. Both the license.json and license.sig files should be backed up.

28.2.3. Operating system backup

It is also necessary to perform appropriate OS backups on a regular basis. However, these activities are outside the scope of this document. Refer to the relevant OS documentation for the appropriate backup procedures.

28.2.4. Kerberos backup

If Kerberos authentication is configured, periodic backups of the Kerberos infrastructure will need to be made. If **mkhpss** was used to configure Kerberos, the Kerberos components that must be backed up are placed in the following locations:

- The Kerberos client configuration file is /etc/krb5.conf.
- The default keytab file is /etc/v5srvtab.
- If a Key Distribution Center (KDC) is running on a node, all files related to it are placed in a single directory: /var/hpss/krb5kdc.

If you configured Kerberos manually, you'll need to determine the file locations yourself. The files that should be backed up are:

- Client configuration file
- KDC configuration file

- KDC acl_file
- KDC admin_keytab
- KDC dict_file
- KDC database_name
- KDC key_stash_file
- KDC default_keytab_name

28.3. HPSS user data recovery: Recovering HPSS files from damaged HPSS volumes

HPSS tape or disk volumes can become unreadable in cases of physical damage to the media or of being accidentally over-written by a non-HPSS application. If an HPSS tape or disk volume is damaged such that HPSS cannot retrieve the user data stored there, its data might be recoverable using the HPSS repack(7) or recover(7) utility.

To determine whether recovery is needed, use this procedure:

1. Determine the name of the potentially damaged volume.

   When HPSS cannot read a disk or tape volume, the HPSS Mover will issue alarm messages which will be displayed in the SSM Alarm and Events window. The alarm messages will contain the name of the physical volume for which the error occurred. Record the volume name.

2. Determine if the volume is actually damaged.

   Check the central HPSS log for the alarm message and any related messages for the source of the problem. Some problems are configuration issues. In particular, for errors for tape volumes, the system administrator should check the block size of the tape device on which the error occurred. HPSS tape devices should be configured with variable block sizes (see Enable variable block sizes for tape devices)

If the problems are not configuration related and the volume does appear to be damaged, repack(7) or recover(7) the suspect volume. See the repack(7) and recover(7) man pages for instructions on running these utilities from the command line.

Repack reads the data from the damaged volume and copies it to a good volume at the damaged level of the hierarchy. **Recover** reads the data from an alternate copy on a volume at another level of the hierarchy and copies it to a good volume at the damaged level of the hierarchy. **Recover** works by staging the data from the alternate copy to level 0 of the hierarchy and forcing the migration of the data all the way down the hierarchy. **Recover** can also be used to clean up the metadata for the damaged volume if the data cannot be restored.

In some situations, one utility may be better suited than the other. In other situations, a combination of the utilities may be best. Some considerations:

- **Recover** requires an alternate copy, which is another complete and permanent copy of the data at another level of the hierarchy. A permanent copy is a copy that is not subject to purge. The

system must be configured in advance to provide two or more alternate copies. If there are no alternate copies of the data, the data cannot be restored by the recover(7) utility. See the recover(7) man page for more information about alternate copies. **Repack** does not require an alternate copy.

- **Repack** must be able to read the damaged volume. If the volume is unreadable, the data cannot be restored by the repack(7) utility. In some cases, repack(7) is able to read part but not all of the data and restore the portions it can read. **Recover** does not read the damaged volume.
- **Recover** will re-migrate the data to all levels of the hierarchy. **Repack** will copy the data only to the required level.
- Since recover(7) stages the data from the alternate copy to level 0 of the hierarchy, there must be sufficient space at level 0 to hold all the alternate file copies. If there is not enough space, the files still might be restored by executing recover(7) in phases. **Recover** can be started and allowed to recover as many files as it is able given the space limitations. Then, once space is freed up at level 0 by normal purge processing, recover(7) can be restarted in "--retry-failures" mode to retry the remaining files. It can be restarted multiple times if necessary until all files are recovered. **Repack** does not depend upon staging the file to level 0 or the amount of space there.
- For tape volumes at any level to which data is migrated from disk, recover(7) by default aggregates the recovered files on the new volume. If for any reason the recovered files cannot be aggregated, recover(7) attempts to place them on the new volume as single files. **Recover** does not aggregate the recovered files if the tape volume is at level 0 or if data is migrated to the volume from tape. **Repack** aggregates files to tape by default, but can be executed with an option not to aggregate. The behavior of repack(7) is the same for tape at any level, regardless of where the tape is in the hierarchy.

If all of the data cannot be restored to another volume by either utility, the recover(7) utility must be used to clean up the metadata for the remaining data so that the volume can be removed from the system.

A good general approach is:

- If you have alternate copies:
  1. Run recover(7) first to try to restore the data from an alternate copy.
  2. If you can't get all the data back from recover(7), run repack(7) to try to get the rest of the data back directly from the damaged volume.
  3. If you still can't get all the data back, run recover(7) again to clean up the metadata for the unrecoverable data. This is a last resort and all other options of recovering the data, including running recover(7) with the **--retry-failures** option, should be attempted first.
- If you don't have alternate copies:
  1. Run repack(7) to try to get as much of the data as possible back directly from the damaged volume.
  2. If you can't get all the data back, run recover(7) to clean up the metadata for the unrecoverable data. This is a last resort and all other options of recovering the data, including running recover(7) with the **--retry-failures** option, should be attempted first.

Once recover(7) is executed on a volume, it must either run to completion or be cancelled. HPSS will not allow the volume to be removed from the system using the "delete resources" operation so long as there is any outstanding recover operation in progress. Even if the recover utility is no longer running and the volume is empty, HPSS will consider the recover operation to be still in progress until it is marked as complete or cancelled. If recover(7) is run for a volume and does not recover all of the files, and repack(7) is then run and does finish retrieving all of the data, the recover operation must be cancelled. See the recover(7) man page for instructions on cancelling the recover operation.



If the first repack attempt fails, it may be worthwhile to retry the repack several times. This is especially true if alternate copies are not available for the storage level that contains the suspect volumes. There are a number of intermittent or environmental errors that may cause an attempt to repack a volume (especially a tape volume) to fail, while a subsequent attempt may succeed (for example, a failing drive, problems reading a volume in a particular drive, or a drive that requires cleaning). If the suspect volume is a tape, try to get it mounted on a number of drives during the repack effort by locking the drives in which it has problems. Sometimes a tape will fail on one drive but can be read on another.

28.4. DB2 monitoring

For the convenience of HPSS administrators, HPSS monitors several critical aspects of the underlying DB2 databases. The amount of free space in the global (often called "cfg") database is monitored by the Core Server running the lowest numbered subsystem. Similarly, the amount of free space in each of the subsystem databases (usually called subsysx, where "x" is the subsystem number) is monitored by the Core Server that runs that subsystem. The configuration settings for these monitors are found in the global and subsystem configurations. The interval at which the monitors run and the thresholds at which they send alarms can be configured to suit the HPSS administrators' needs.

Starting with HPSS version 6.2.2, a DB2 Log Monitor is available in HPSS. This monitor periodically checks the DB2 transaction log files to make sure the primary and mirrored logs are congruent and apparently performing normally. It also scans the DB2 diagnostic log, `db2diag.log`, for indications of any sort of problem with the transaction logs. If any errors are reported in `db2diag.log`, the HPSS DB2 Log Monitor sends an alarm and indicates the file path and line number where the error was reported. The administrator should be able to open `db2diag.log` and examine the problem report in detail. The error is reported one time for each occurrence in `db2diag.log`. This monitoring function does not interfere with normal maintenance of `db2diag.log`. When the file is trimmed back, or removed, the monitor automatically tracks these changes.

HPSS provides a DB2 diagnostics tool that uses Db2's monitoring routine tables to gather metrics about client connections, database activity, and table-level operations. This tool provides a real-time snapshot of DB2's health, making it easier to identify issues and maintain performance. For a comprehensive breakdown, refer to the man page hpss_db_diag(7).

28.5. DB2 space shortage

As HPSS adds files to the database, the DB2 container spaces will be consumed. If the DB2 Space Monitor is configured to monitor free space, an alarm will be generated when free space falls below a set threshold. The following sections discuss how to handle the appropriate space shortage issue.

Before performing any of the suggestions or steps listed in this section, consult the appropriate *DB2 Administration Guide: Implementation* for the appropriate version of DB2. It would also be wise to verify any significant changes to, or reallocation of, production DB2 resources with HPSS support.

28.5.1. DMS table spaces

Capacity for a DMS table space is the total size of containers allocated to the table space. When a DMS table space reaches capacity (depending on the usage of the table space, 90% is a possible threshold), you should add more space to it. The database manager will automatically rebalance the tables in the DMS table space across all available containers. During rebalancing, data in the table space remains accessible.

A DB2 container may contain more than one physical disk, but a physical disk can be used in only one DB2 container. We recommend allocating the entire physical disk to a DB2 container and not leaving over any free space on the physical disk since it will be considered wasted space once the disk is allocated to the DB2 container.

The *DB2 Administration Guide: Implementation* states that you can increase the size of a DMS table space by adding one or more containers to the table space.

To add a container to a DMS table space using the Control Center:

1. Expand the object tree until you see the Table Spaces folder.
2. Right-click the table space you wish to add the container to, and select Alter from the pop-up menu.
3. Select the Containers tab, click Add, complete the information, and click OK.
4. Click Ok.

To add a container to a DMS table space using the command line, enter:

```
ALTER TABLESPACE <name> ADD (DEVICE '<path>' <size>)
```

For example:

```
ALTER TABLESPACE CFG ADD (DEVICE '/dev/rhd5' 10000)
```

Note that size is in pages based on the page size of the existing table space you are adding the container too. In this case, container refers to the DEVICE.

Note that the ALTER TABLESPACE statement allows you to change other properties of the table space that can affect performance. Refer to "Table Space Impact on Query Optimization" in the *DB2 Administration Guide: Performance* for more information.

28.5.2. SMS table spaces

In general, you cannot extend the size of an SMS table space very easily because SMS capacity depends on the space available in the file system and the maximum size of the file supported by the operating system. You should be able to increase the size of a file system using the normal operating system facilities for increasing file system space.

Chapter 29. Management tools

29.1. Utility overview

HPSS provides a variety of command-line utility programs to aid in the use and management of the system. The programs can be grouped into the following major categories:

29.1.1. Fileset and junction management

These programs work for HPSS filesets.

crtjunction(7)

Creates an HPSS junction.

deljunction(7)

Deletes an HPSS junction.

lsfilesets(7)

Lists all HPSS filesets.

lsjunctions(7)

Lists all HPSS junctions.

29.1.2. Disk management

These programs manage operations on disk in HPSS.

rehome_files(7)

Rewrite files on disk with an updated storage segment size based on the current disk storage class configuration.

29.1.3. Tape management

These programs manage the life cycle of tapes in the system.

repack(7)

Consolidates data on disks and tapes by moving storage segments from sparsely populated volumes to other volumes. The emptied volumes are then available for reuse or removal from the system. This is the principal tool for recovering data from degraded RAIT volumes and writing it to new volumes.

reclaim(7)

Reinitializes an empty tape for reuse after repack has emptied it, or after it has become empty through attrition or repacking. Reclaim will not reinitialize a tape volume that has been retired, or degraded RAIT volumes.

retire(7)

Marks a disk or tape volume as read-only and not reclaimable.

remove(7)

Deletes empty disk and tape volumes from the Core Server's metadata, then optionally exports the volumes from the PVL/PVR.

recover(7)

Salvages data from damaged volumes by reading from other copies or other levels in the hierarchy.

lbp_verify(7)

Mounts tape volumes and verifies the data against stored CRCs.

These tools deal with shelving tapes:

shelf_tape(7)

Selects and moves tapes from a tape library to an offline storage area. Additionally, shelf_tape can check in tapes that previously existed on shelf.

shelf_tape_util(7)

Performs various administrative functions related to shelved tapes.

29.1.4. System info

dump_sspvs(7)

Lists the physical volumes known to a particular subsystem, including statistics about file aggregates on tapes and RAIT volume status.

dumpbf(7)

Lists the storage characteristics of a particular bitfile.

lsvol(7)

Lists the path names of bitfiles that are located on a disk or tape volume.

dumppv_pvl(7)

List the physical volumes managed by the PVL.

dumppv_pvr(7)

List the physical volumes managed by a particular PVR.

hpss_db_metrics(7)

Query the HPSS system for metrics from the HPSS database, including purge-locked data, trashcan data, unmigrated data, and unpurged data.

hpss_server_metrics(7)

Query the HPSS system for metrics from HPSS servers. This includes metrics from movers, storage, configuration, limits, core server interfaces, trashcans, and drives.

lshpss(7)

Lists various views of the system configuration.

lsrb(7)

Lists a list of bitfiles that have not been accessed since a given date/time.

lspvhist(7)

List physical—as opposed to virtual—tape volume history information.

mps_reporter(7)

Produces human-readable output from MPS summary files.

plu(7)

Lists files that have a purge record in HPSS metadata. Can be used to see which bitfiles have been purge locked (for example, bitfiles that cannot be purged).

lscos(7)

Lists all bitfile COS change records by subsystem. It also allows an administrator to cancel all COS change requests for a particular subsystem, and print a summary of COS change records by stream.

device_scan(7)

Lists information all SCSI-attached devices including media-changers, disks, tape.

scsi_home(7)

Displays SCSI home locations.

29.1.5. System management

hpssadm(7)

Allows administrators to perform some administration of HPSS without the graphical user interface. Not all SSM functionality is available. But it enables the administrator to perform all configuration operations, to display alarms and events, to display all the lists provided by the SSM GUI, and to manage HPSS servers, devices, storage classes, PVL jobs, volumes, filesets, and junctions. It may be run interactively or in batch mode.

hpssgui(7)

Starts the Graphical User Interface program which is the primary administrative interface for HPSS.

hacl

Allows viewing and editing of HPSS access control lists (ACLs).

hpssuser(7)

Adds, manages, and removes HPSS users.

rtmu(7)

(Real-Time Monitoring Utility) Displays the state of requests in the HPSS system.

hpss_avaledit(7)

Allows HPSS administrators at sites using SITE-style accounting and account validation to edit the relationships between user IDs, account IDs, and account names.

hpss_errno(7)

Translate HPSS error codes to descriptive strings.

hpss_server_acl(7)

Updates authorization data in the database.

hpss_krb5_keytab

Generates a keytab in the form usable by the hpssadm(7) program.

hpss_krb5_rmxcred

Removes expired cached Kerberos credentials from the default credential directory.

hpss_unix_group(7)

Manages HPSS UNIX group file.

hpss_unix_keygen

Create an HPSS UNIX master key.

hpss_unix_keytab(7)

A keytab generation program for the HPSS UNIX native security service.

hpss_unix_passwd(7)

Changes the HPSS UNIX native security service password for an HPSS user.

hpss_unix_user(7)

Manages HPSS UNIX password file.

hpss_managetables

Creates, deletes and modifies databases, database tablespaces, tables, views and constraints. It should be used by hand only by extremely knowledgeable administrators. This program can destroy database tables.

xmladdindex(7)

Creates an XML index.

xmldeleteindex(7)

Deletes an XML index.

schema_registrar(7)

Registers an XML schema with the database.

hpsssum(7)

Checksums HPSS or local files

rq_cleanup(7)

Clean up persisted request queue entries.

29.1.6. User interfaces

pftp_client(7)

Used to perform parallel file transfers between a local and remote host.

29.1.6.1. HPSS UNIX Tools

The HPSS unix tools are client API tools which mimic basic behaviors familiar to system users such as chmod, chown, cp, etc. These tools all use the same authentication method, which can be defined in the user's environment, to authenticate via kerberos or UNIX keytab.

This set of tools has minimal setup or configuration requirements and can be easy to use as a basic suite to interact with HPSS for administrators or simple workflows.

One downside of these tools is that they will authenticate every time they run, unlike other applications such as HSI which allow multiple operations per session. This can place a heavy load on authentication services depending on how the tools are used. Many of these tools can operate on multiple files at a time, which can mitigate this.

hpsscat(7)

Similar to the cat tool.

hpsschmod(7)

Similar to the chmod tool.

hpsschown(7)

Similar to the chown tool.

hpsscp(7)

Copy files to and from HPSS.

hpssgetvol(7)

Get storage volume information for a file.

hpssgetxattrs(7)

Get extended attributes for a file.

hpsslimits(7)

Get HPSS core server limits.

hpssln(7)

Create HPSS hard and soft links.

hpssls(7)

List files in HPSS.

hpssmkdir(7)

Make directories in HPSS.

hpsspurge(7)

Purge files in HPSS.

hpsspurgelock(7)

Purge-lock files in HPSS.

hpssrm(7)

Remove files in HPSS.

hpssrmdir(7)

Remove directories in HPSS.

hpsssetcos(7)

Change the COS of a file in HPSS.

hpssstat(7)

Stat a file in HPSS.

hpsstouch(7)

Similar to the touch tool.

hpssxfind(7)

Find files in HPSS based on user-defined attributes.

29.1.7. User-defined Attributes

schema_registrar(7)

Register an XML schema which controls the attributes and values allowed by a User-defined Attributes function.

xmladdindex(7)

Add an index to the User-defined Attributes to enable faster searches of attribute paths.

xmldeleteindex(7)

Delete an index on the User-defined Attributes.

29.1.8. Testing and debugging

scrub(7)

A general-purpose HPSS test shell; a driver for the Client API library. It can perform various data operations, such as reading and writing undifferentiated data, migrating and purging specific files, and browsing the HPSS name space. **Scrub** also supports doing some simple operations with user-defined attributes. It does not transfer external files into or out of HPSS.



Scrub is a test tool used by developers for testing the Client API, Core Server, and

> *other areas of HPSS. As such, it does not provide many of the protections offered by other interfaces into the system. For this reason, it must be treated by production sites with caution. It is available for limited use by production sites for information gathering and for other operations described in official HPSS documentation. To prevent unintended consequences, any other use at production sites should be cleared with HPSS support.*

29.1.9. Unsupported tools

Tools in this category are not formally supported. They are provided as-is and are found in the `tools/unsupported` portion of the HPSS source tree.

disk_allocation_stat

Calculates and displays statistics on disk usage and fragmentation for disk storage classes.

ftp

(DISCOM PFTP Daemon) This is a portable version of a standard UNIX FTP daemon which supports the HPSS Parallel FTP protocols. By replacing the System FTP Daemon and configuring the DISCOM Daemon, significant improvements can be made in moving data between two UNIX-based systems. Both a Kerberized version and a non-Kerberized version are available for a number of hardware/software platforms. The HPSS Parallel FTP Client should be used to talk to this server. Do not confuse this with the HPSS FTP daemon.

hpssstats

Reports HPSS statistics.

Load tools

These utilities reside in `tools/unsupported/src/load`. Each tool allows the manual loading, editing and deletion of one or more metadata records.

magic

Allows direct manipulation of name space objects.

nsde

Displays and edits Name Service database tables.

PVrr

Displays the number of PV retrieval requests that have been made on a volume.

removesegs

Searches for and removes disk and tape storage segments that are not pointed to by any bitfile metadata. It requires at least two separate searches of the bitfile metadata over a period of time before it will remove orphaned segments. It can also be used to reset the number of segments counter in disk and tape storage maps and to repair certain BFS metadata defects.

showdiskmaps

Sends a command to the Core Server in the selected storage subsystem to dump its in-memory disk space allocation maps to a file in `${HPSS_PATH_TMP}`, then displays that information on

standard output. The tool's output can give a point-in-time snapshot of how space is allocated across the disk VVs in a system and how busy those disk VVs (and the movers that operate them) are. The output includes information about free, usable, and actual space; number of I/O operations in flight; and the number and nature of blocks available for allocation.

29.2. Tape queue monitoring

HPSS supports dumping out tape queue data. When read, write, dismount, or copy requests are made to tape resources, they are queued up and ordered using the Core Server's configured ordering method.

Periodically, the entire tape queue layer's status is dumped to local disk in order to facilitate administrative queue monitoring. The location of this dump is `$HPSS_PATH_TMP/tape-schedule-<subsystem id>.json` (for example, `/var/hpss/tmp/tape-schedule-1.json`). The file is in JSON format and is overwritten every five seconds (by default). The dump interval can be modified by setting the environment variable HPSS_SCHEDULE_DUMP_INTERVAL_SECONDS and restarting the Core Server. It is not recommended to decrease this interval below five seconds. This dump remains enabled so long as the path can be opened for writing. These fields are subject to change between HPSS releases.

29.2.1. Example


```
{
 "VV": [
  {
   "Queue": {
    "Status": "Active",
    "Sessions": [
     {
      "Activity Id": 0,
      "Start Section": 28,
      "Is Complete": false,
      "TimeOrder": 0,
      "Score": 0,
      "Request Id": 1003421707,
      "Is Reader": true,
      "Status": "RUNNING",
      "Start Offset": 0,
      "Wait Time": 95745,
      "Data Length": 1024,
      "Is Dismount": false,
      "Is Copy Pair": false,
      "Is Writer": false
     }
    ]
   },
   "Processing I/O": 0,
   "RAO Size": 0,
   "Minimum Processing Time": 0,
```

```
    "Sorting Type": "Section Reorder",
    "VVID": "00000001-04-00000003-01e46b88d6979fa4-3c73",
    "Mounted PV Name": "E0100100",
    "PV Name": "E0100100",
    "Mounted Device Id": 0,
    "Last Write": 1438876024,
    "Last Read": 1438876024,
    "Max Active Set": 1024,
    "Total Requests Scheduled": 0,
    "Active Set": 0,
    "Last Preference Change": 1438876024,
    "Write Preference Aid": 0,
    "Last Used": 95,
    "Number Scheduling Errors": 0,
    "Min Wait Time": 0,
    "Preference": "WRITERS",
    "I/O Efficiency": 0.0,
    "Average Wait Time": 0,
    "Max Wait Time": 0,
    "Average Schedule Time": 0,
    "Media Speed": 80,
    "Average Seek": 20000,
    "Number of Schedules": 0,
    "Schedule Time": 0,
    "Average Schedule": 10000,
    "Number Unscheduled": 0,
    "Average Processing Time": 0,
    "Maximum Processing Time": 0
   }
  ]
}
```

29.2.2. JSON field descriptions

VV

The top level type is an ordered list, each containing statistics about a VV, its queue, and its sessions. The following member fields are defined:

Average Processing Time

The average time to process a request, in milliseconds.

VVID

The unique VV identifier

Average Seek

The average seek time, in milliseconds, for this media type. This is a static value set in the **Average Latency** field for the storage class.

Mounted PV Name

The cartridge label of the Mounted Volume. In a multi-stripe or RAIT VV, the "Mounted Volume" is the name of the volume which is being sent RAO requests.

Minimum Processing Time

The minimum time required to process a request, in milliseconds.

Number Scheduling Errors

The number of consecutive times a scheduling request has resulted in an error.

Average Schedule Time

The average time required to run the scheduler.

Schedule Time

Total time spent scheduling, in milliseconds.

Active Set

The currently active set number. Items which are not in the same set are never scheduled together; items in the lowest set are always scheduled first. The active set identifier is assigned to a request as requests enter the scheduler.

Max Active Set

The size of an active set. For example, when "1024" is specified up to 1024 requests will be scheduled together. Any items in addition to that 1024 will reside in a different active set, and will be scheduled after all items in the current set are complete.

PV Name

The first PV in the VV. This may not be the "Mounted PV" for multi-stripe VVs, but it will be how the VV is referenced in other places in the system.

Mounted Device Id

The device ID where the "Mounted PV" is mounted. If the PV is not mounted, this is zero.

Average Wait Time

Average time a request has been waiting before it is allowed to run, in milliseconds.

Sorting Type

The method used by the scheduler to order the requests. Valid values:

- `FIFO` : First in, First Out
- `Section Reorder` : Based upon the tape section and offset data
- `RAO` : Based upon an RAO (Recommended Access Order) request to the Mounted Device.

Last Used

Seconds since the last time the VV was used.

Last Write

Seconds since epoch of the last time a write was completed. Initially this will be the time

when the queue was created.

Preference

The current queuing preference. Valid values:

- `WRITERS` indicates writers are preferred. This is the default value.
- `READERS` indicates readers are preferred. This preference may occur when readers have been waiting on a writer for a substantial amount of time.

I/O Efficiency

The number of megabytes per second transferred, compared to the media type's stated megabytes per second.



This value is an estimate only and is biased low due to including mount and positioning times.

Maximum Processing Time

The maximum amount of time spent processing a request.

Last Read

Seconds since epoch of the last time a read was completed. Initially this will be the time when the queue was created.

Total Requests Scheduled

The total number of requests scheduled for this queue's lifetime. Note this is not a persistent value across mount requests, system outages, and so on.

Write Preference Aid

The activity identifier for the request which caused the queue to go into read preference mode. This allows the readers to run, but also allows the preference to switch to being write-preferenced after all long-running readers have completed.

Last Preference Change

Seconds since epoch of the last time the queuing preference changed.

Media Speed

The speed of the media, in megabytes per second. This is based upon the storage class.

Processing I/O

Total data processed divided by the total processing time.

RAO Size

Maximum RAO request size. For Section Ordered configurations which utilize RAO-capable devices, this represents the maximum number of requests that can be scheduled together (even if the Active Set is configured much higher). This is based upon the parameters of the media and is not configurable.

Min Wait Time

Minimum time a request has been waiting before it is allowed to run, in milliseconds.

Average Schedule

Average seek time, in milliseconds, when scheduling. This is currently statically defined as half of the storage class media's *average* latency.

Number of Schedules

Number of times the scheduler has been run.

Number Unscheduled

Number of requests waiting which have not yet been scheduled.

Queue

VV Queue Object containing details of requests.

Queue

Each VV has a Queue field, which contains details regarding the queue requests. The following member fields are defined:

Status

The queue's activity state. Valid values are:

- `Active` indicates the queue's session data is active.
- `Currently being rescored` indicates the queue's session data cannot be displayed since it is being rescored.

Sessions

An ordered list (ordered by Score) of all sessions in the queue and their details.

Sessions

Each request is associated with a session. The session data includes details about the request.

Start Section

The starting tape section for the request.

TimeOrder

The order of the request, in FIFO terms.

Score

The order of the request, based upon the Core Server (or VV's) ordering type.

Status

The status of the session. Valid values are:

- `RUNNING` indicates this request is currently running.
- `WAITING` indicates this request is currently waiting.

Activity Id

The unique activity ID for this session within the VV's queue.

Request Id

The request ID for this request.

Is Copy Pair

Whether this request belongs to a **copy pair** (usually a tape repack or tape-to-tape migration).

Wait Time

How long the request has been waiting, in milliseconds.

Data Length

Length of the data being read or written.

Start Offset

Starting section offset.

Is Complete

Whether the request is complete.

Is Reader

Whether the request is for a read.

Is Dismount

Whether the request is to dismount the tape.

Is Writer

Whether the request is a writer.

29.3. HPSS ACL permission exceptions

The purpose of this section is to describe the minor differences between the HPSS ACL implementation and the ACL implementation described in the *DFS Administration Guide*. The HPSS ACL implementation was modeled after the implementation described in the *DFS Administration Guide*. And with the exception of the d (delete) and i (insert) permissions, the HPSS ACL implementation is a duplication of the DFS implementation.

In the HPSS ACL implementation the d and i permissions can be set and deleted on directory objects (d and i permissions do not apply to files), but the existence (or absence) of d and i is ignored when determining whether or not a user has the permissions necessary to perform an operation.

For example HPSS allows objects to be created in a directory even if the i (insert) permission is not set on that directory. Likewise HPSS allows objects to be deleted from a directory even if that directory does not have the d (delete) permission set. In all other regards the HPSS ACL implementation mimics the DFS implementation.

Table 5 in the ACL Permission section of the *DFS Administration Guide* describes the various

operations that can be performed on files or directories and the ACL permissions that are required to perform these operations. Table 5 is reproduced here with the d (delete) and i (insert) permissions removed. The following table describes how HPSS performs these operations and the ACL permissions required to perform them.

Operation	**Required permissions**
Change to a directory	**Execute (x)** on the directory itself. **Execute (x)** on all directories that lead to the directory.
List the contents of a directory	**Read (r)** on the directory itself. **Execute (x)** on all directories that lead to the directory.
List information about the objects in a directory	**Read (r)** and **execute (x)** on the directory itself. **Execute (x)** on all directories that lead to the directory.
Create an object	**Write (w)**, **execute (x)** on the directory in which the object is to be placed. **Execute (x)** on all directories that lead to the directory in which the object is to be placed.
Delete an object	**Write (w)**, **execute (x)** on the directory from which the object is to be deleted. **Execute (x)** on all directories that lead to the directory from which the object is to be deleted.
Rename an object	**Write (w)**, **execute (x)** on the object's current directory. **Execute (x)** on all directories that lead to the object's current directory. **Write (w)**, **execute (x)** on the object's new directory. **Execute (x)** on all directories that lead to the object's new directory. **Write (w)** on the object if the object is a directory.
Read or read lock a file	**Read (r)** on the file itself. **Execute (x)** on all directories that lead to the file.
Write or write lock a file	**Write (w)** on the file itself. **Execute (x)** on all directories that lead to the file.
Execute a binary file	**Execute (x)** on the file itself. **Execute (x)** on all directories that lead to the file.
Execute a shell script	**Read (r)** and **execute (x)** on the script itself. **Execute (x)** on all directories that lead to the script.
List the ACLs on an object	**Execute (x)** on all directories that lead to the object.

Change the ACLs on an object	**Control (c)** on the object itself. **Execute (x)** on all directories that lead to the object.

Chapter 30. HPSS features

The following section describes selected HPSS features.

30.1. End to End Data Integrity

End to End Data Integrity is supported by HPSS by using file hashing and tape logical block protection.

End to End Data Integrity properly starts in the application writing data into HPSS. HPSS supports associating a checksum with HPSS files. HPSS will validate that checksum during migration to tape. If a checksum is not provided, HPSS can generate checksums during migration depending on the class of service configuration.

Logical block protection allows HPSS to submit CRCs for each block being written to the tape drive which the tape drive can use to validate the data the tape drive received against what the HPSS tape Mover was processing. When the LBP is not enabled, the tape drive calulates its own CRC during the write, which allows the drive to verify that is has read what it orignally wrote, but it is not guaranteed that it was sent by HPSS. HPSS processes such as Repack will automatically verify the CRC data as HPSS moves data between tapes over its lifetime.

The HPSS tape validation Tool, `lbp_verify`, validates the data integrity of HPSS tape volumes by requesting the tape verify each active part of the tape against the block protection CRCs. During `lbp_verify`, no data is moved off the tape drive; all of the processing and validation is done by the tape and reported back by HPSS.



Tapes which were imported prior to enabling LBP on the storage class do not have LBP enabled. They must be re-imported to begin using the LBP feature.

30.1.1. How to Enable LBP

Navigate to the desired Tape Storage Class Configuration. Select "Block Protection" and update the configuration.

30.1.2. How to Enable File Hash

To enable automatic checksumming by HPSS during migration, navigate to the desired Class of Service Configuration. Select the desired checksum (note that the higher bit size SHAs have slower performance - make sure that your desired checksum performs at your tape drive rate), and update the configuration. Note that checksumming during migration has some limitations, which include that the tape storage class is single striped.



Don’t forget to apply your changes using the Apply Config action.

30.1.3. Check your volumes with lbp_verify

Periodically verify your tapes with `lbp_verify`. This process is similar to reading all of the active

data on the tape and can take many hours to complete.

```
# Check out the options for lbp_verify
$ /opt/hpss/bin/lbp_verify -help
# Verify the volume containing PV name '000718'
$ lbp_verify --subsystem-id=1 -v 000718 Verify volume '000718'
Complete: Volume: 00071800 Sections: 1001 Blocks: 207531 Errors: 0
```

30.2. HPSS Config History

HPSS supports retrieving the history of configuration changes. This feature can help an administrator understand changes to the system configuration over time, and when changes were applied to the system.

The primary means of interacting with HPSS config history is through the new `hpss_cfg_history` tool. This tool displays the changes to the system from a given timeframe in JSON format. As an example, this will show the changes since October 30, 2024 at 16:32:11.

```
%/opt/hpss/bin/hpss_cfg_history -a -T 2024-10-30-16.32.11
{"since": "2024-10-30-16.32.11", "core": {"5": {"MAX_ACTIVE_IO_REQS": {"old": 4990,
"new": 100}}}, "server": {"Core Server": {"REQUEST_QUEUE_SIZE": {"old": 2500, "new":
1000}, "THREAD_POOL_SIZE": {"old": 5000, "new": 1000}, "MAX_CONNECTIONS": {"old":
5000, "new": 2000}}}}
```

The output here has the type (core), followed by the ID (server ID 5), followed by the fields that have been changed. Note that some information (e.g. core server changes) are split between two definitions - one is the basic server configuration under "Core" and the other is under "Server". The output shows the old and new values. By default, if no time is provided with -T, the tool will show changes in the last 24 hours.

In the example, changes for all tables are shown, which is done via the "-a" option. If an admin just wanted to see the changes to the server table, he could replace "-a" with "-l server". If he just wanted to see the server and core tables, he could instead use "-l server,core".

See the `hpss_cfg_history` page for more details.

30.3. Encryption in HPSS

HPSS supports three types of encryption to secure data and communications:

30.3.1. Client Data Encryption

Client data encryption protects user data during transfers between HPSS clients and the storage system. When enabled, HPSS generates an ephemeral key and negotiates encryption as part of the transfer request.

30.3.1.1. Enabling Client Data Encryption

Client data encryption is disabled by default. To enable it, set `HPSS_API_ENCRYPT=1` in the client's `/var/hpss/etc/env.conf` file.

Additional configuration options:

- **HPSS_API_CIPHER** - Specifies the cipher to use for data transfers. Run `hpss_list_enc enc` to list available ciphers. Note that configuration policy can restrict which ciphers are allowed.
- **HPSS_API_KEY_LEN** - Sets the encryption key size (default: 32 bytes/256 bits). The key is hashed through a key expansion algorithm to produce the required bytes for the cipher.

See Client API configuration for complete details on these environment variables.

30.3.1.2. Client-Specific Encryption Setup

Different HPSS clients have specific ways to enable encryption:

- **PFTP (Parallel FTP)** -
  - **Session-based encryption** - Use the `PKEY` command to set the cipher and encryption key for the current session before initiating parallel transfers (`PRTR` for reads, `PSTO` for writes). The cipher and key are maintained for the session duration and reset when the file is closed.
  - **Secure Authentication with data protection** - Configure PFTP to use "Secure Authentication" (HPSS.conf) and set the data protection level to "private" using the `PROT` command. This enables encryption for all data transfers in the session.

    See the PFTP documentation in the *HPSS User's Guide* for details on these commands.
- **S3 API Proxy** - Configure the `hpsspftp.cipher.name` setting. See [s3-hpss-encrytion] for details on S3 encryption configuration, including cipher selection, key generation rounds, and buffer sizes.
- **HPSSFS-FUSE** - Use the `encrypt` mount option or set the `HPSS_API_ENCRYPT` environment variable. See the *HPSSFS-FUSE Administrator's Guide* for mount option details.
- **Other Client Applications** - Consult the specific application documentation. Consult the *HPSS Programmer's Reference* for API-level encryption configuration.

30.3.2. Mover Data Encryption

Mover data encryption secures the control channel communications between Movers and the Core Server, as well as between Movers and clients.

30.3.2.1. Configuring Mover Encryption

Mover encryption requires configuration in multiple locations:

1. **Mover Cipher Type** - Set in `/var/hpss/etc/site.conf` on each Mover system. To list available ciphers, run: `hpss_list_enc enc` If no cipher is specified, mover control communication will not be encrypted. See the `site.conf` man page for details.
2. **Mover Encryption Key** - Configure in the Mover-specific configuration window. Each Mover

requires a unique encryption key to secure its interface with the Core Server.

3. **Encryption Key Files** - The encryption key must be stored in a file accessible from the Mover's local file system (e.g., `/var/hpss/etc/mvr_ek`).

30.3.2.2. Settings Propagation

For Mover encryption to work correctly:

- The cipher specified in `site.conf` must be available on both the Mover system and any client systems that will connect to it (verify with `hpss_list_enc enc`)
- The encryption key configured in the Mover's specific configuration must match the key in the encryption key file

See Additional Mover configuration for complete Mover encryption setup details.

30.3.3. Control Traffic Encryption

Control traffic encryption secures RPC (Remote Procedure Call) communications between HPSS servers and clients. This is distinct from data encryption and focuses on protecting control information and authentication data.

30.3.3.1. RPC Protection Levels

HPSS supports multiple RPC protection levels, configured via the **Packet Security** setting in server configurations:

- **Connect** - Authenticates the connection but provides no data protection
- **Packet** - Authenticates each RPC packet
- **Packet Integrity** - Verifies that data has not been modified during transmission
- **Packet Privacy** - Encrypts all data transferred between client and server (highest security, highest overhead)

Higher protection levels require more processing for each RPC. See Security controls for server-specific RPC protection configuration.

30.3.3.2. Client RPC Configuration

HPSS provides a default RPC protection level via `HPSS_RPC_PROT_LEVEL` in `/var/hpss/etc/env.conf`. Individual client interfaces can override this default using their own specific environment variables:

- **HPSS_API_RPC_PROT_LEVEL** - For Client API applications
- **HPSSGUI_RPC_PROT_LEVEL** - For HPSS GUI (hpssgui) interface
- **HPSSADM_RPC_PROT_LEVEL** - For HPSS Admin (hpssadm) command-line tool

The SSM GUI and ADM can also be controlled via command line and via ssm.conf. See SSM configuration file for details.

Valid values for all RPC protection level variables are:

- `connect` - Authenticates the connection only
- `pkt`(Java) or `packet` (API) - Authenticates each RPC packet
- `pkt_integ` (Java) or `packet integrity` (API) - Verifies data integrity
- `pkt_privacy` (Java) or `packet privacy` (API) - Encrypts all RPC data

If not explicitly set, these variables default to the value of `HPSS_RPC_PROT_LEVEL`. Note that the HPSSGUI and HPSSADM use the underscore format (e.g., `pkt_integ`) while the API uses space-separated format (e.g., `packet integrity`).

See Client API configuration for complete client RPC configuration details.

30.3.3.3. Web-Based Management Interface Encryption

The SSM API (SSMAPI) and Web SSM (WebSSM) interfaces use HTTPS to encrypt all communications between the web browser and the API server. The SSMAPI communicates with the SSM System Manager using the configured RPC protection level settings defined in the server's interface configuration.

30.3.3.4. Additional Security Configuration

Control traffic encryption is also configured through:

- **Server Interface Configuration** - Each server type has specific interface controls for RPC security. See Security controls for details.



Control traffic encryption (RPC security) and data encryption are independent features that can be configured separately based on your security requirements.

Appendix A: Glossary of terms and acronyms

ACL	Access Control List
ACSLS	Automated Cartridge System Library Software (Oracle StorageTek)
ADIC	Advanced Digital Information Corporation
accounting	The process of tracking system usage per user, possibly for the purposes of charging for that usage. Also, a log record type used to log accounting information.
ACI	AML Client Interface
AIX	Advanced Interactive Executive. An operating system provided on many IBM machines.
alarm	A log record type used to report situations that require administrator investigation or intervention.
AML	Automated Media Library. A tape robot.
AMS	Archive Management Unit
ANSI	American National Standards Institute
API	Application Program Interface
archive	One or more interconnected storage systems of the same architecture.
ASLR	Address Space Layout Randomization
attribute	When referring to a managed object, an attribute is one discrete piece of information, or set of related information, within that object.
attribute change	When referring to a managed object, an attribute change is the modification of an object attribute. This event may result in a notification being sent to SSM, if SSM is currently registered for that attribute.
audit (security)	An operation that produces lists of HPSS log messages whose record type is SECURITY. A security audit is used to provide a trail of security-relevant activity in HPSS.

AV	Account Validation
bar code	An array of rectangular bars and spaces in a predetermined pattern which represent alphanumeric information in a machine-readable format (such as a UPC symbol).
BFS	HPSS Bitfile Service
bitfile	A file stored in HPSS, represented as a logical string of bits unrestricted in size or internal structure. HPSS imposes a size limitation in 8-bit bytes based upon the maximum size in bytes that can be represented by a 64-bit unsigned integer.
bitfile segment	An internal metadata structure, not normally visible, used by the Core Server to map contiguous pieces of a bitfile to underlying storage.
Bitfile Service	Portion of the HPSS Core Server that provides a logical abstraction of bitfiles to its clients.
BBTM	Blocks Between Tape Marks. The number of data blocks that are written to a tape virtual volume before a tape mark is required on the physical media.
CAP	Cartridge Access Port
cartridge	A physical media container, such as a tape reel or cassette, capable of being mounted on and dismounted from a drive. A fixed disk is technically considered to be a cartridge because it meets this definition and can be logically mounted and dismounted.
class	A type definition in Java. It defines a template on which objects with similar characteristics can be built, and includes variables and methods specific to the class.
Class of Service	A set of storage system characteristics used to group bitfiles with similar logical characteristics and performance requirements together. A Class of Service is supported by an underlying hierarchy of storage classes.
cluster	The unit of storage space allocation on HPSS disks. The smallest amount of disk space that can be allocated from a virtual volume is a cluster. The size of the cluster on any given disk volume is determined by the size of the smallest storage segment that will be allocated on the volume, and other factors.

configuration	The process of initializing or modifying various parameters affecting the behavior of an HPSS server or infrastructure service.
COS	Class of Service
control path	For the SCSI PVR, this is a connection to the library for sending commands. Control paths can be discovered using device_scan.
Core Server	An HPSS server which manages the name space and storage for an HPSS system. The Core Server manages the name space in which files are defined, the attributes of the files, and the storage media on which the files are stored. The Core Server is the central server of an HPSS system. Each storage subsystem uses exactly one Core Server.
CRC	Cyclic Redundancy Check
CS	Core Server
daemon	A UNIX program that runs continuously in the background.
DAS	Distributed AML Server
DB2	A relational database system, a product of IBM Corporation, used by HPSS to store and manage HPSS system metadata.
DCE	Distributed Computing Environment
debug	A log record type used to report internal events that can be helpful in troubleshooting the system.
delog	The process of extracting, formatting, and outputting HPSS central log records. This process is obsolete in 7.4 and later versions of HPSS. HPSS logs are now recorded as plain text.
deregistration	The process of disabling notification to SSM for a particular attribute change.
descriptive name	A human-readable name for an HPSS server.
device	A physical piece of hardware, usually associated with a drive, that is capable of reading or writing data.
directory	An HPSS object that can contain files, symbolic links, hard links, and other directories.

dismount	An operation in which a cartridge is either physically or logically removed from a device, rendering it unreadable and unwritable. In the case of tape cartridges, a dismount operation is a physical operation. In the case of a fixed disk unit, a dismount is a logical operation.
DNS	Domain Name Service
DOE	Department of Energy
DPF	Database Partitioning Feature
drive	A physical piece of hardware capable of reading or writing mounted cartridges. The terms device and drive are often used interchangeably.
EB	Exabyte (2^{60})
EOF	End of File
EOM	End of Media
ERA	Extended Registry Attribute
ESCON	Enterprise System Connection
event	A log record type used to report informational messages (for example, subsystem starting or subsystem terminating).
export	An operation in which a cartridge and its associated storage space are removed from the HPSS system Physical Volume Library. It may or may not include an eject, which is the removal of the cartridge from its Physical Volume Repository.
FC SAN	Fiber Channel Storage Area Network
FIFO	First in first out
file	An object than can be written to, read from, or both, with attributes including access permissions and type, as defined by POSIX (P1003.1-1990). HPSS supports only regular files.
file family	An attribute of an HPSS file that is used to group a set of files on a common set of tape virtual volumes.
fileset	A collection of related files that are organized into a single easily managed unit. A fileset is a disjoint directory tree that can be mounted in some other directory tree to make it accessible to users.
fileset ID	A 64-bit number that uniquely identifies a fileset.

fileset name	A name that uniquely identifies a fileset.
file system ID	A 32-bit number that uniquely identifies an aggregate.
FTP	File Transfer Protocol
FSF	Forward Space File
FSR	Forward Space Record
Gatekeeper	An HPSS server that provides two main services: the ability to schedule the use of HPSS resources referred to as the Gatekeeping Service, and the ability to validate user accounts referred to as the Account Validation Service.
Gatekeeping Service	A registered interface in the Gatekeeper that provides a site the mechanism to create local policy on how to throttle or deny create, open and stage requests and which of these request types to monitor.
Gatekeeping Site Interface	The APIs of the gatekeeping site policy code.
Gatekeeping Site Policy	The gatekeeping shared library code written by the site to monitor and throttle create, open, and/or stage requests.
GB	Gigabyte (2^{30})
GECOS	The comment field in a UNIX password entry that can contain general information about a user, such as office or phone number.
GID	Group Identifier
GK	Gatekeeper
GSS	Generic Security Service
GUI	Graphical User Interface
HA	High Availability
HACMP	High Availability Clustered Multi-Processing - A software package used to implement high availability systems.
HADR	DB2 High Availability Disaster Recovery
halt	A forced shutdown of an HPSS server.
HBA	Host Bus Adapter
HDM	Shorthand for HPSS/DMAP.

hierarchy	See *storage hierarchy*.
HPSS	High Performance Storage System
HPSS-only fileset	An HPSS fileset that is not linked to an external file system (such as XFS).
HTP	HPSS Test Plan
IBM	International Business Machines Corporation
ID	Identifier
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronics Engineers
import	An operation in which a cartridge and its associated storage space are made available to the HPSS system. An import requires that the cartridge has been physically introduced into a Physical Volume Repository (injected). Importing the cartridge makes it known to the Physical Volume Library.
I/O	Input/Output
IOD/IOR	I/O Descriptor/I/O Reply. Structures used to send control information about data movement requests in HPSS and about the success or failure of the requests.
IP	Internet Protocol
IRIX	SGI's implementation of UNIX
JRE	Java Runtime Environment
junction	A mount point for an HPSS fileset.
KB	Kilobyte (2^{10})
KDC	Key Distribution Center
LAN	Local Area Network
LANL	Los Alamos National Laboratory
latency	For tape media, the average time in seconds between the start of a read or write request and the time when the drive actually begins reading or writing the tape.
LBP	Logical Block Protection
LDAP	Lightweight Directory Access Protocol

LFT	Local File Transfer
LLNL	Lawrence Livermore National Laboratory
LMU	Library Management Unit
Location Service	A module within the HPSS Core Server used to help clients locate the appropriate Core Server and/or other HPSS server to use for a particular request.
log record	A message generated by an HPSS application and handled and recorded by the HPSS logging subsystem.
log record type	A log record may be of type alarm, event, info, debug, request, security, trace, or accounting.
logging service	An HPSS infrastructure service consisting of the logging subsystem and one or more logging policies. A default logging policy can be specified, which will apply to all servers, or server-specific logging policies may be defined.
LS	Location Service
LSM	Library Storage Module
LTO	Linear Tape-Open. A half-inch open tape technology developed by IBM, HP, and Seagate.
LUN	Logical Unit Number
LVM	Logical Volume Manager
MAC	Mandatory Access Control
managed object	A programming data structure that represents an HPSS system resource. The resource can be monitored and controlled by operations on the managed object. Managed objects in HPSS are used to represent servers, drives, storage media, jobs, and other resources.
MB	Megabyte (2^{20})
MBS	Media Block Size
metadata	Control information about the data stored under HPSS, such as location, access times, permissions, and storage policies. Most HPSS metadata is stored in a DB2 relational database.
method	A Java function or subroutine.

migrate	To copy file data from a level in the file's hierarchy onto the next lower level in the hierarchy.
Migration/Purge Server	An HPSS server responsible for supervising the placement of data in the storage hierarchies based upon site-defined migration and purge policies.
MM	Metadata Manager. A software library that provides a programming API to interface HPSS servers with the DB2 programming environment.
mount	An operation in which a cartridge is either physically or logically made readable/writable on a drive. In the case of tape cartridges, a mount operation is a physical operation. In the case of a fixed disk unit, a mount is a logical operation.
mount point	A place where a fileset is mounted in the XFS and HPSS name spaces.
Mover	An HPSS server that provides control of storage devices and data transfers within HPSS.
MPS	Migration/Purge Server
MVR	Mover
NASA	National Aeronautics and Space Administration
Name Service	The portion of the Core Server that provides a mapping between names and machine-oriented identifiers. In addition, the Name Service performs access verification and provides the Portable Operating System Interface (POSIX).
name space	The set of name-object pairs managed by the HPSS Core Server.
NERSC	National Energy Research Supercomputer Center
NIS	Network Information Service
NLS	National Language Support
notification	A notice from one server to another about a noteworthy occurrence. HPSS notifications include notices sent from other servers to SSM of changes in managed object attributes, changes in tape mount information, and log messages of type alarm or event.
NS	HPSS Name Service
NSL	National Storage Laboratory

object	See *managed object*.
ORNL	Oak Ridge National Laboratory
OS	Operating System
OS/2	The operating system (multi-tasking, single user) used on the AMU controller PC.
PB	Petabyte (2^{50})
PFTP	Parallel File Transfer Protocol
PFTPD	PFTP Daemon
physical volume	An HPSS object managed jointly by the Core Server and the Physical Volume Library that represents the portion of a virtual volume. A virtual volume may be composed of one or more physical volumes, but a physical volume may contain data from no more than one virtual volume.
Physical Volume Library	An HPSS server that manages mounts and dismounts of HPSS physical volumes.
Physical Volume Repository	An HPSS server that manages the robotic agent responsible for mounting and dismounting cartridges or interfaces with the human agent responsible for mounting and dismounting cartridges.
PIO	Parallel I/O
PIOFS	Parallel I/O File System
POSIX	Portable Operating System Interface (for computer environments).
purge	Deletion of file data from a level in the file's hierarchy after the data has been duplicated at lower levels in the hierarchy and is no longer needed at the deletion level.
purge lock	A lock applied to a bitfile which prohibits the bitfile from being purged.
PV	Physical Volume
PVL	Physical Volume Library
PVM	Physical Volume Manager
PVR	Physical Volume Repository

RAID	Redundant Array of Independent Disks
RAIT	Redundant Array of Independent Tapes
RAM	Random Access Memory
RAO	Recommended Access Order
reclaim	The act of making previously written but now empty tape virtual volumes available for reuse. Reclaimed tape virtual volumes are assigned a new Virtual Volume ID, but retain the rest of their previous characteristics. Reclaim is also the name of the utility program that performs this task.
registration	The process by which SSM requests notification of changes to specified attributes of a managed object.
reinitialization	An HPSS SSM administrative operation that directs an HPSS server to reread its latest configuration information, and to change its operating parameters to match that configuration, without going through a server shutdown and restart.
repack	The act of moving data from a virtual volume onto another virtual volume with the same characteristics with the intention of removing all data from the source virtual volume. Repack is also the name of the utility program that performs this task.
request	A log record type used to report some action being performed by an HPSS server on behalf of a client.
RISC	Reduced Instruction Set Computer/Cycles
RPC	Remote Procedure Call
RSF	Reverse Space File
RSR	Reverse Space Record
SCSI	Small Computer Systems Interface
security	A log record type used to report security-related events (for example, authorization failures).
SGI	Silicon Graphics
shelf tape	A cartridge which has been physically removed from a tape library but whose file metadata still resides in HPSS.
shutdown	An HPSS SSM administrative operation that causes a server to stop its execution gracefully.

sink	The set of destinations to which data is sent during a data transfer, such as disk devices, memory buffers, or network addresses.
SM	System Manager
SMC	SCSI Medium Changer
SME	Subject Matter Expert
SNL	Sandia National Laboratories
SOID	Storage Object ID. An internal HPSS storage object identifier that uniquely identifies a storage resource. The SOID contains a unique identifier for the object, and a unique identifier for the server that manages the object.
source	The set of origins from which data is received during a data transfer, such as disk devices, memory buffers, or network addresses.
SP	Scalable Processor
SS	HPSS Storage Service
SSD	Solid State Drive
SSH	Secure Shell
SSI	Storage Server Interface
SSM	Storage System Management
SSM session	The environment in which an SSM user interacts with the SSM System Manager to monitor and control HPSS. This environment may be the graphical user interface provided by the **hpssgui** program, or it may be the command-line user interface provided by the **hpssadm** program.
SSMSM	Storage System Management System Manager
stage	To copy file data from a level in the file's hierarchy onto the top level in the hierarchy.
start-up	An HPSS SSM administrative operation that causes a server to begin execution.
info	A log record type used to report file staging and other kinds of information.
STK	Storage Technology Corporation (Oracle StorageTek)

storage class	An HPSS object used to group storage media together to provide storage for HPSS data with specific characteristics. The characteristics are both physical and logical.
storage hierarchy	An ordered collection of storage classes. The hierarchy consists of a fixed number of storage levels numbered from level 1 to the number of levels in the hierarchy, with the maximum level being limited to 5 by HPSS. Each level is associated with a specific storage class. Migration and stage commands result in data being copied between different storage levels in the hierarchy. Each Class of Service has an associated hierarchy.
storage level	The relative position of a single storage class in a storage hierarchy. For example, if a storage class is at the top of a hierarchy, the storage level is 1.
storage map	An HPSS object managed by the Core Server to keep track of allocated storage space.
storage segment	An HPSS object managed by the Core Server to provide abstract storage for a bitfile or parts of a bitfile.
Storage Service	The portion of the Core Server which provides control over a hierarchy of virtual and physical storage resources.
storage subsystem	A portion of the HPSS name space that is managed by an independent Core Server and (optionally) Migration/Purge Server.
Storage System Management	An HPSS component that provides monitoring and control of HPSS via a windowed operator interface or command-line interface.
stripe length	The number of bytes that must be written to span all the physical storage media (physical volumes) that are grouped together to form the logical storage media (virtual volume). The stripe length equals the virtual volume block size multiplied by the number of physical volumes in the stripe group (that is, stripe width).
stripe width	The number of physical volumes grouped together to represent a virtual volume.
System Manager	The Storage System Management (SSM) server. It communicates with all other HPSS components requiring monitoring or control. It also communicates with the SSM graphical user interface (**hpssgui**) and command line interface (**hpssadm**).

TB	Terabyte (2^{40})
TCP/IP	Transmission Control Protocol/Internet Protocol
TDS	Tivoli Directory Server
TI-RPC	Transport-Independent-Remote Procedure Call
trace	A log record type used to record procedure entry/exit events during HPSS server software operation.
transaction	A programming construct that enables multiple data operations to possess the following properties: • All operations commit or abort/roll-back together such that they form a single unit of work. • All data modified as part of the same transaction are guaranteed to maintain a consistent state whether the transaction is aborted or committed. • Data modified from one transaction are isolated from other transactions until the transaction is either committed or aborted. • Once the transaction commits, all changes to data are guaranteed to be permanent.
TSA/MP	Tivoli System Automation for Multiplatforms
TSM	Tivoli Storage Manager
UDA	User-defined Attribute
UDP	User Datagram Protocol
UID	User Identifier
UPC	Universal Product Code
UUID	Universal Unique Identifier
VPN	Virtual Private Network
virtual volume	An HPSS object managed by the Core Server that is used to represent logical media. A virtual volume is made up of a group of physical storage media (a stripe group of physical volumes).
virtual volume block size	The size of the block of data bytes that is written to each physical volume of a striped virtual volume before switching to the next physical volume.

VV	Virtual Volume
XDSM	The Open Group's Data Storage Management standard. It defines APIs that use events to notify Data Management applications about operations on files.
XFS	A file system created by SGI available as open source for the Linux operating system.
XML	Extensible Markup Language
OIDC	OpenID Connect, an authentication protocol built on top of the OAuth 2.0 authorization framework. It enables applications to verify the identity of a user based on authentication performed by an identity provider and to obtain basic profile information about the user in a standardized way.
OIDC provider	An OpenID Connect (OIDC) provider (IdP) such as Azure AD, Okta, Keycloak, or other enterprise identity management solutions.

Appendix B: References

1. ***3580 Ultrium Tape Drive Setup, Operator and Service Guide*** GA32-0415-00
2. ***3584 UltraScalable Tape Library Planning and Operator Guide*** GA32-0408-01
3. ***3584 UltraScalable Tape Library SCSI Reference*** WB1108-00
4. ***AIX Performance Tuning Guide***
5. ***Data Storage Management (XDSM)*** API, ISBN 1-85912-190-X
6. ***HACMP for AIX, Version 4.4: Concepts and Facilities***
7. ***HACMP for AIX, Version 4.4: Planning Guide***
8. ***HACMP for AIX, Version 4.4: Installation Guide***
9. ***HACMP for AIX, Version 4.4: Administration Guide***
10. ***HACMP for AIX, Version 4.4: Troubleshooting Guide***
11. ***HPSS Error Messages Reference Manual***, current release
12. ***HPSS Programmer's Reference***, current release
13. ***HPSS Programmer's Reference - I/O Supplement***, current release
14. ***HPSS User's Guide***, current release
15. ***IBM SCSI Device Drivers: Installation and User's Guide***, GC35-0154-01
16. ***IBM Ultrium Device Drivers Installation and User's Guide*** GA32-0430-00.1
17. ***IBM Ultrium Device Drivers Programming Reference*** WB1304-01
18. ***Interfacing Guide DAS***, Order no. DOC F00 011
19. ***Installing, Managing, and Using the IBM AIX Parallel I/O File System***, SH34-6065-02
20. ***Platform Notes: The hme FastEthernet Device Driver*** 805-4449
21. ***POSIX 1003.1-1990 Tar Standard***
22. ***Reference Guide AMU***, Order no. DOC E00 005
23. ***STK Automated Cartridge System Library Software (ACSLS) System Administrator's Guide***, PN 16716
24. ***STK Automated Cartridge System Library Software Programmer's Guide***, PN 16718
25. J. Steiner, C. Neuman, and J. Schiller, "***Kerberos: An Authentication Service for Open Network Systems***," USENIX 1988 Winter Conference Proceedings (1988).
26. R.W. Watson and R.A. Coyne, "***The Parallel I/O Architecture of the High-Performance Storage System (HPSS)***," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.
27. T.W. Tyler and D.S. Fisher, "***Using Distributed OLTP Technology in a High-Performance Storage System***," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

28. J.K. Deutsch and M.R. Gary, "***Physical Volume Library Deadlock Avoidance in a Striped Media Environment***," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

29. R. Grossman, X. Qin, W. Xu, H. Hulen, and T. Tyler, "***An Architecture for a Scalable, High-Performance Digital Library***," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

30. S. Louis and R.D. Burris, "***Management Issues for High-Performance Storage Systems***," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

31. D. Fisher, J. Sobolewski, and T. Tyler, "***Distributed Metadata Management in the High Performance Storage System***," from the 1st IEEE Metadata Conference, April 16-18, 1996.

Appendix C: Developer acknowledgments

HPSS is a product of a government-industry collaboration. The project approach is based on the premise that no single company, government laboratory, or research organization has the ability to confront all of the system-level issues that must be resolved for significant advancement in high-performance storage system technology.

HPSS development was performed jointly by IBM Worldwide Government Industry, Lawrence Berkeley National Laboratory, Lawrence Livermore National Laboratory, Los Alamos National Laboratory, NASA Langley Research Center, Oak Ridge National Laboratory, and Sandia National Laboratories.

We would like to acknowledge Argonne National Laboratory, the National Center for Atmospheric Research, and Pacific Northwest Laboratory for their help with initial requirements reviews.

We also wish to acknowledge Cornell Information Technologies of Cornell University for providing assistance with naming service and transaction management evaluations and for joint developments of the Name Service.

In addition, we wish to acknowledge the many discussions, design ideas, implementation and operation experiences we have shared with colleagues at the National Storage Laboratory, the IEEE Mass Storage Systems and Technology Technical Committee, the IEEE Storage System Standards Working Group, and the storage community at large.

We also wish to acknowledge the Cornell Theory Center and the Maui High Performance Computer Center for providing a test bed for the initial HPSS release.

We also wish to acknowledge Gleicher Enterprises, LLC for the development of the HSI, HTAR, and Transfer Agent client applications.

Finally, we wish to acknowledge CEA-DAM (**Commissariat à l'Énergie Atomique - Centre d'Études de Bruyères-le-Châtel**) for providing assistance with development of NFS V3 protocol support.

Appendix D: HPSS.conf configuration file

The HPSS.conf(7) configuration file contains tuning options to be used by HPSS clients and servers.

General `HPSS.conf` *rules and suggestions:*

- Keywords *must* be specified precisely as shown (no extra spaces). All items are case-sensitive.
- Lines are comprised of *comments*, *blank lines*, *simple specifiers*, *specifiers* and *values* - "abc = def", or *compound specifiers* and *terminators* - "def = \{ ...}."
- A semicolon (";") is used to comment out (deactivate) an actual configuration option. To activate these options, remove the semicolon. (Suggestion)
- Statements with a pound sign ("#") sign as the first non-white character are explanatory comments. (Suggestion)
- Only ten levels of specification are allowed: Stanzas, SubStanzas, Sections, and SubSections.
- SubStanzas may only exist in Compound Stanzas. Sections may only exist in Compound SubStanzas, and SubSections may only exist in Compound Sections.
- No line may exceed 512 characters, including the "= \{".
- Comments may be included by entering either a semicolon (";") or a pound sign ("#") as the first non-white character in a line. Use of either of these characters other than as the first character will not be interpreted as a comment (rather, it will be interpreted as part of the specifier or value). This means you can not put comments at the end of a line.
- Indentation is optional but is strongly encouraged (assists in diagnosis). Use "tabs" for indentation. (Strong suggestion)
- Closing braces ("}") must be used to terminate opening braces (" = \{"). This *must* appear on a separate line. Maintaining indentation is recommended.
- Spaces before or after the equal sign ("=") are optional.
- Blank lines are ignored.
- If the default value works, don't fix it. (Strong suggestion)
- The non-HPSS PFTP daemon options should be left alone. (Suggestion)



The uncommented settings in `HPSS.conf.tmpl` *should be suitable for a "vanilla" UNIX/UNIX HPSS system. The exceptions are "PFTP Client Interfaces" and "Network Interfaces" Stanzas. HPSS and network tuning are highly dependent on the application environment. The values specified herein are* not *expected to be applicable to any installation. These two Stanzas should be tailored to suit your site configuration.*

D.1. PFTP Client Stanza

The PFTP client configuration options are in two distinct Stanzas of the `HPSS.conf` file (PFTP Client Stanza and PFTP Client Interfaces Stanza).

Table 33. PFTP Client Stanza fields

Configuration type	Abbreviated description
Stanza (CMPD)	PFTP Client = \{ Example: `PFTP Client = \{` *Optional Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza	SYSLOG Facility = <value> Values: **DAEMON**, **LOCAL0** ... **LOCAL7** Example: `SYSLOG Facility = LOCAL2` *Optional* SubStanza specifying the Syslog Facility for the Multinode Daemon
SubStanza	Debug Value = <value> Values: 0 - 3 Example: `Debug Value = 1` *Optional* SubStanza specifying the Level of Debugging for the PFTP client. A larger number provides more information.
SubStanza	Protocol = <value> Values: PDATA_AND_MOVER, PDATA_ONLY (default), PDATA_PUSH Example: `Protocol = PDATA_AND_MOVER` *Optional* SubStanza specifying the default protocol. May contain any of the three protocols supported.
SubStanza	Auto Parallel Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation Example: `Auto Parallel Size = 4MB` *Optional* SubStanza specifying the minimum file size to start using the "auto-parallel" features of the PFTP client.

Configuration type	Abbreviated description
SubStanza	PortRange = <value> Value: **StartPort-EndPort** Example: `PortRange = 10100-12100` *Optional* SubStanza specifying the TCP port range to use between the PFTP client and the Mover(s). This may be necessary when port range filters are used for security. Note: The old format (ncacn_ip_tcp[10100-12100]) is still supported for now, but will be phased out soon.
SubStanza	Parallel Block Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation Example: `Parallel Block Size = 512KB` *Optional* SubStanza specifying the size of the data blocks to be used for parallel data transfers
SubStanza	Transfer Buffer Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation Example: `Transfer Buffer Size = 16MB` *Optional* SubStanza specifying the PFTP client I/O buffer sizes
SubStanza	Socket Buffer Size = <value> Value: Viable socket size specified as a decimal number or "*x*MB" style notation Example: `Socket Buffer Size = 1MB` *Optional* SubStanza specifying the Pdata socket buffer sizes

Configuration type	Abbreviated description
SubStanza	MAX Ptran Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation. NOLIMIT - Set to 250GB, the absolute max. Example: `MAX Ptran Size = 10GB` *Optional* SubStanza specifying a larger transfer size between socket open and closure. For disk COSs, the segment sizes may potentially override this specification.
SubStanza	Enable SAN3P Example: `Enable SAN3P` *Optional* Enable SAN3P support. Default is SAN3P disabled. Note: this will also set the protocol to PDATA_AND_MOVER.
SubStanza	TA API Debug Level Values: 0 - 4 Example: `TA API Debug Level = 0` *Optional* SubStanza specifying the level of debugging for the transfer agent. A larger number provides more information.
SubStanza	TA API Debug Log Value: Debug log file path Example: `TA API Debug Log = /var/TA/logs/PFTPC_TA_API_debug_%N_%P.log` *Optional* SubStanza specifying the path to the transfer agent log file. A **%N** will be replaced by the number of agents, and a **%P** will be replaced by the PID of the particular agent.
SubStanza	Special Features Enabled Example: `Special Features Enabled` *Optional* SubStanza for performance testing *only*. Should *not* be activated except by the appropriate personnel. Default is Off.

Configuration type	Abbreviated description
SubStanza	Disable stagebatch on mget Example: `Disable stagebatch on mget` *Optional* SubStanza specifying disabling using the batch stage site command to stage files prior to calling mget. Enabled unless this is active.
SubStanza	Use DirectIO Example: `Use DirectIO` *Optional* Enable Direct I/O for reads from the local file system. This is beneficial in cases where kernel caching may hinder performance.
SubStanza	DirectIO Block Size Example: `DirectIO Block Size = 512` *Optional* The block size of the local disk. Direct I/O requires that the transfer be block-aligned with the local device.

Note: All PFTP Client SubStanzas are optional.

The **PFTP Client** = **\{ ... }** Stanza contains several optional specifications for the pftp_client(7) executables.

The **SYSLOG Facility** = **value** is used to establish the syslog facility for the multinode daemon. It has no relevance to the PFTP client running without the multinode daemon.

The **Debug Value** = **value** is used to provide additional diagnostic information for the PFTP client. Under normal circumstances, this should be set to "0" (default) or "1". Larger values will provide additional information, but will cause users to complain.

The **Secure Authentication** = **value** tells the client whether or not to expect to securely authenticate. Unless the value is "required" or "attempted", no secure authentication will be attempted even if Kerberos is set up.

The **Secure Authentication Mechanism** = **value** tells the client which secure authentication mechanism to use. The default is Kerberos. The possible values are KRB5 for Kerberos or UNIX for HPSS UNIX.

The **Protocol** = **value** SubStanza is used to specify the desired PFTP protocol. Currently, any of three values may be specified: **PDATA_AND_MOVER**, **PDATA_ONLY,** or **PDATA_PUSH**. The default specification is **PDATA_AND_MOVER**. The **PDATA_ONLY** specification provides improved performance in high latency WAN environments.

The pftp_client automatically performs conversion of **get** and **put** commands to their parallel equivalents, **pget** and **pput**. Some sites have reported degraded performance as a result of this

substitution occurring with small file transfers. To accommodate this problem, the **Auto Parallel Size** = **value** SubStanza may be specified in the `HPSS.conf` file where the "automatic" parallel features will *not* be invoked if the size of the file is less than the value provided. The value may be specified as a decimal number (1048576) or in the format: *x*MB.

For sites with "Firewalls/ Diodes" installed, it may be necessary to provide specific ports for the data to move between the PFTP client and the Mover(s). This can be accomplished by specifying the **PortRange** = **value** SubStanza. The syntax of this statement is:

```
PortRange = 10100-12100
```

The old syntax of the value (ncacn_ip_tcp[10100-12100]) identical to DCE's **RPC_RESTRICTED_PORTS** environment variable is still supported for now, but will be phased out soon. At this time, the restricted ports are ignored for passive transfers. Arbitrary ports will be assigned.

Additional options are available for controlling the size of the PFTP transfer buffers, **Transfer Buffer Size**, and the buffer size for the sockets in the **PDATA_ONLY** protocol, **Socket Buffer Size**. The value may be specified as a decimal number (such as "1048576") or in the format: *x*MB.

The PFTP parallel protocol opens and closes the sockets between the PFTP client child processes and any Movers. The default value for tape was every 512 MB and for disk was the smaller of the size of 64 storage segments or 512 MB. With transfers increasing in performance into the 100 MB/second and greater range, the opening and closing of sockets is another performance problem. The **MAX Ptran Size** = **value** SubStanza has been provided to allow for larger transfers between socket open and closing.



in the case of disks, the 64 storage segments is still the overriding specification, so storage classes need careful specification to provide very large segments if the value associated with **MAX Ptran Size** is large. An artificial limit of 250 GB is compiled into the PFTP client, which should not cause a great concern any time in the near future. Even at 1 GB/second, this is several minutes. The value may be specified as a decimal number (such as "4294967296") or in the format: *x*GB.

Under normal operating conditions, the **Special Features Enabled** component should remain commented out.

PFTP Client Stanza example (with suggested contents):

```
PFTP Client = {
    # SYSLOG facility to use for all syslog messages by the Multinode Daemon.
    SYSLOG Facility = LOCAL0
    # Debugging Levels
    ; Debug Value = 1
    # Set Default Protocol
    # Values are PDATA_ONLY, PDATA_AND_MOVER, or PDATA_PUSH
    # Default is PDATA_ONLY
    ; Protocol = PDATA_AND_MOVER
```

```
    # Set Minimum File Size for auto mapping Non-parallel
    # commands to Parallel commands
    Auto Parallel Size = 4MB
    # The Port Range to be used for connections to other machines
    # Useful if Client is used to cross Network Filters
    # (The older ncacn_ip_tcp[10100-12100] format is supported for now)
      # Default = 10100-65535
    ; PortRange = 10100-12100
    # PDATA Options
    ; Parallel Block Size = 512KB
    Transfer Buffer Size = 16MB
    ; Socket Buffer Size = 1MB
    # PFTP sets an Artificial (Compiled in) Maximum of 250GB
    MAX Ptran Size = 10GB
    # Enable SAN3P
    # Note: this will also set protocol to PDATA_AND_MOVER
    ; Enable SAN3P
    # Location of temp file for parallel pipes
    # Default = /tmp
    ; Temp Pipe File Path = /tmp
    # The (optional) Agent Debug Level Stanza sets the
    # debug level (0-4, 0=no debug, 4=max debug) for the
    # Transfer Agent processes.  This overrides the command
    # line "-d" option for the agent, if specified in this file.
    TA API Debug Level = 0
    # The Debug Logfile contains debugging output. It can
    # be overridden by environment variable settings.
    ; TA API Debug Log = /var/TA/logs/PFTPC_TA_API_debug_%N_%P.log
    # Special Features
    ; Special Features Enabled
}
```

D.2. PFTP Client Interfaces Stanza

Many systems have multiple interfaces, some of which may not have access to all destination hosts. The PFTP Client Interfaces Stanza is used to specify which interfaces on the source host should be used to communicate to destination PFTP daemons, HPSS Movers, or both. This is particularly useful if both low-speed and high-speed interfaces are available to the client host and the PFTP data transfers should use the high-speed interfaces.

Table 34. PFTP Client Interfaces Stanza fields

Configuration type	Abbreviated description
Stanza (CMPD)	PFTP Client Interfaces = \{ Example: `PFTP Client Interfaces = \{` *Optional Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza (CMPD)	<Hostname> <hostname.domain> = \{ Example: `my_host my_host.my.domain = \{` Contains the hostname(s) executing the PFTP client. Must be terminated with a matching "}" Multiple Hostname SubStanzas may be in a single `HPSS.conf` file representing multiple PFTP client hosts sharing the `HPSS.conf` file
Section (CMPD)	<Name> <Name> = \{ Name: One or more hostnames Example: `storage storage.my.domain = \{` Contains the hostname(s) executing the PFTP daemon. Must be terminated with a matching "}" Multiple Daemon Sections may be in a single Hostname SubStanza representing multiple PFTP daemon destinations which may use different characteristics
SubSection	<Name> or <Dotted IP address> Name: **Valid Interface Name** Dotted IP address: **132.175.1.1** Example: `eth0` Example: `132.175.1.1` *Optional* parameter containing the name or dot notation IP address specification for the interface on the local host (**PFTP client**) to use to connect to the Mover(s) associated with the specified PFTP daemon

The **PFTP Client Interfaces = \{ ... }** Stanza contains several configuration options for the pftp_client(7) executables.

SubStanzas refer to the hostname(s) associated with the local system where the pftp_client is being invoked.

Sections refer to the PFTP daemon hostname(s) where the PFTP daemon is being invoked.

SubSections refer to the network interface to be utilized (by the host where the PFTP client is invoked) to transfer data to the HPSS Mover systems.



*For HPSS, this is **not** necessarily the name of the Mover(s).*

SubSections specify names or dot notation IP addresses of the interfaces on the local host to be used. For HPSS, all of these interfaces must be able to connect to the Mover(s).



If and only if a specific COS is specified, these interfaces need only provide connection to the Mover(s) associated with the specific COS.

PFTP Client Interfaces Stanza rules:

- Source hostnames may contain one or more hostnames separated by white spaces (subject to the `HPSS.conf` line character limit).
- "Default" is a reserved notation to be used if the local host is not in the Stanza.
- Destination Host (FTPD Host) may contain one or more hostnames separated by white spaces (subject to the `HPSS.conf` line character limit).
- Interface Specification must be specified by interface name or IP Address dot notation.
- Interfaces ***must*** be able to connect to destination (HPSS Mover).



Communication failures that are not easily diagnosed will occur if the interface specification is invalid.

The following example is completely commented out. The default interface(s) will be used. This is probably typical for many sites and does not need to be modified unless multiple interfaces and asymmetric networks are involved.

PFTP Client Interfaces Stanza example:

```
; PFTP Client Interfaces = {
   # PFTP Client Host Name(s)
   ; water.clearlake.ibm.com water = {
      # Next Specification is the PFTP Daemon Host
      # water has 3 specific interfaces that can talk
      # to the HPSS Movers associated with the PFTP
      # Daemon Host "water", as well as various
      # interfaces of the form 192.2.nnn.nnn
      ; water.clearlake.ibm.com water = {
         # Interfaces on the host specified as the Client Machine
         ; 192.94.47.227
```

```
         ; 192.175.14.35
         ; 192.222.197.1
         ; eth*
      ; }
      # water has ONLY 1 interface that can talk to the HPSS
      # Movers associated with the PFTP Daemon Host "sneezy"
      ; sneezy sneezy.clearlake.ibm.com = {
         ; 192.94.47.227
      ; }
      # Use the default water interface, 192.100.101.1, to talk
      # to any other PFTP Daemons.
      ; Default = {
         ; 192.100.101.1
      ; }
   ; }

   ; sneezy sneezy.clearlake.ibm.com = {
      ; larry larry.clearlake.ibm.com = {
         ; 192.94.47.226
      ; }
      ; sneezy sneezy.clearlake.ibm.com = {
         ; 192.94.47.226
      ; }
   ; }

   # For all other Client Hosts - This allows a single HPSS.conf
   # file to be available using a common files system. This is
   # ONLY useful for cluster systems that specify "Common"
   # interfaces for multiple # nodes of the cluster (I/O Nodes)
   ; Default = {
      # Client Host Name
      ; water water.clearlake.ibm.com = {
         ; 134.253.14.227
      ; }
   ; }
; }
```

D.3. Multinode Table Stanza

The HPSS PFTP client normally forks children to provide multiple network paths between the PFTP client and any Movers. In some instances, it may be preferable to have these processes (pseudo children) running on independent nodes. In this case, it is necessary to set up a multinoded(7) daemon on the independent node or host and have the PFTP client initiate one or more data transfer processes with these child processes. The **Multinode Table** Stanza is used to specify what remote hosts are to perform the "pseudo" PFTP client child processes functions.

Table 35. Multinode Table Stanza fields

Configuration type	Description
Stanza (CMPD)	Multinode Table = \{ Example: `Multinode Table = \{` *Optional Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza	Sleep for Debugger = values Value: Time in seconds Example: `Sleep for Debugger = 15` *Optional* parameter to specify a delay in the multinode daemons to allow diagnosis. This should *only* be specified for diagnostics and will unnecessarily cause degradation if misused. Leave commented out.
SubStanza (CMPD)	<Local Hostname(s)> Example: `my_host my_host.my.domain = \{` Contains the local hostname(s) this SubStanza represents Must be terminated with a matching "}"
Section (CMPD)	<Remote Hostname(s)> Example: `FTP_host PFTP_host.domain = \{` Contains the hostname(s) of the system running the PFTP server Must be terminated with a matching "}"

Configuration type	Description
Sub-Section	<remote_host> or <remote_host> = <dot notation interface> Example: `his_name` Example: `his_name = 100.102.103.45` Contains the hostname in either string format or dot notation IP address of the host to act as a "pseudo" PFTP child. If a secondary name is specified after the "=", the first interface is to be used for the "control" connection between the PFTP client and the multinoded hosts and the second specification is the interface to be used for the "data" connection(s) to the Mover(s). If only one value is provided, it represents BOTH the "control" and "data" connections.

The **Multinode Table** = **\{ ... }** Stanza contains one or more SubStanzas specifying the names of the host initiating the PFTP session.

Each Section contains one or more names or IP addresses of remote hosts executing a multinode daemon (multinoded(7)). The remote host must have appropriate entries for the **inetd** or **xinetd** superdaemon (`/etc/inetd.conf` and `/etc/services`) to initiate the multinoded.

The Sections may be either a simple Section or a valued Section. A simple SubStanza is a single name or dot notation IP address to be used for both "control" connection and "data" connection. The valued SubStanza is used to specify the name or dot notation IP address for the "control" connection (specifier) and the name or dot notation IP address for the "data" connection (value).

Multinode Table Stanza rules:

- **SubStanza** hostnames (local hosts) may contain one or more hostnames separated by white spaces (subject to the `HPSS.conf` line character limit.)
- **Section** hostnames (remote hosts) [and/or values] may be specified as either string-based hostnames or dot notation IP addresses. Only one entry per line.

Multinode Table example:

```
# Options read by the Multinode Daemon
Multinode Table = {
    # Diagnostic Delay
    ;  Sleep for Debugger = 15
    # Hostname of the Client
    water water.clearlake.ibm.com =
       # Name of the system running the PFTP Server
       pftp_server pftp_server.clearlake.ibm.com = {
```

```
            # Specification of the Multinode Host
            # If the Data Node is a different interface than the interface
            # specified by the Control Node; then enter the Data Node
            # Interface after the "=" otherwise the Control and Data
            # are the same.
            # Control and/or Data may be dot notation OR string hostnames.
            Water = 134.253.14.227
        }
    }
    Default = {
        Default = {
            # If the Data Node is different than the Control Node
            # Enter the Data Node after the "=" otherwise the
            # Control and Data are the same.
            # Control and/or Data may be dot notation OR string hostnames.
            larry = sneezy
        }
    }
}
```

D.4. Network Options Stanza

The network options are in the **Network Options** = **\{ ... }** Stanza of the `HPSS.conf` file.

The **Network Options** Stanza allows different options to be specified based on source IP address [Local Interface Name(s)] and destination IP address(es). When the PFTP client, Client API, or Mover establish connections, they will search the contents of this file for entries matching source and destination IP addresses and use the options specified in the matching entry.

The configuration file entries contain values and flags to be used for applying assorted socket and network options including: whether to enable or disable TCP Delay **(TcpNoDelay)**, the socket send sizes **(SendSpace)** and/or socket receive sizes **(RecvSpace)**, the desired write buffer size **(WriteSize)**, and **RFC1323** support ("Large" Window).

Configuring the TCP socket buffer sizes may also require changing parameters in the OS to allow for larger buffer sizes. TCP socket buffer sizes should be configured based on the client's **bandwidth delay product**, that is, the link capacity (BW) multiplied by the ping time (RTT) should be smaller than the TCP socket buffer size divided by the ping time.

```
BW * RTT <= TCP / RTT
```

Which configuration entry to use is determined based on the Local Interface Name and the destination IP address "masked" by the **NetMask** value. The calling application (PFTP client, Client API, or Mover) will apply the value of the NetMask specification in the configuration file entry to the specified destination address. A **"Default"** destination may be specified for all sources and destinations not explicitly specified in the `HPSS.conf` file.

In Linux, there are several sysctl parameters that can limit the ability of HPSS to change socket

buffer sizes. These include **net.core.rmem_max**, **net.core.wmem_max**, **net.ipv4.tcp_rmem**, and **net.ipv4.tcp_wmem** which can be reviewed and modified using sysctl. The `HPSS.conf` settings control what HPSS will attempt to set socket buffer sizes to based on network address, but will be limited to the range supported by the kernel.

Table 36. Network Options Stanza fields

Configuration type	Description
Stanza (CMPD)	Network Options = \{ Example: `Network Options = \{` *Optional Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza	Default Write Size = <value> Example: `Default Receive = 1MB` *Optional* SubStanza specifying the default network Read socket size if not specified explicitly
SubStanza	Default Write Size = <value> Example: `Default Send Size = 1MB` *Optional* SubStanza specifying the default network write socket size if not specified explicitly
SubStanza	Default Write Size = <value> Example: `Default Write Size = 4MB` *Optional* SubStanza specifying the default write size if not specified explicitly
SubStanza (CMPD)	<Source IP Interface Name> = \{ Example: `my_host my_host.my.domain = \{` *Optional* SubStanza specifying an interface name. May contain one or more names separated by white spaces. May contain: "Default = \{" (*Reserved* Specification) for inclusion of entries not explicitly specified Must be terminated with a matching "}"

Configuration type	Description
Section (CMPD)	<Destination IP address> = \{ Example: `100.101.102.0 = \{` *Optional* SubStanza specifying a dotted decimal address of the destination interface Only one address is allowed; however, networks and sub-networks may be chosen by the appropriate specification of the NetMask. May contain: "Default = \{" (*Reserved* Specification) for inclusion of entries not explicitly specified Must be terminated with a matching "}"
SubSection	NetMask = <value> Value: Viable netmask as IP address Example: `NetMask = 255.255.255.0` *Optional* parameter to specify the dotted decimal net mask to apply to the destination IP address to determine whether the entry applies
SubSection	RFC1323 = <value> Values: 0, 1 Example: `RFC1323 = 1` *Optional* parameter to specify whether the RFC1323 option should be disabled ("0") or enabled (any other value)
SubSection	SendSpace = <value> Values: Size specified as decimal value or "xMB" format Example: `SendSpace = 256KB` *Optional* parameter to specify the value to be used for the socket sending buffer space

Configuration type	Description
SubSection	RecvSpace = <value> Values: Size specified as decimal value or "xMB" format Example: `RecvSpace = 256KB` *Optional* parameter to specify the value to be used for the socket receive buffer space
SubSection	WriteSize = <value> Values: Size specified as decimal value or "xMB" format Example: `WriteSize = 1MB` *Optional* parameter used to set the size to be used for each individual write request to the network
SubSection	TcpNoDelay = <value> Values: 0, 1 Example: `TcpNoDelay = 1` *Optional* parameter Indicates whether the TCP Delay option should be disabled (0) or enabled (any other value)

SendSpace and **RecvSpace**. Controls the size of the receive and send buffers for TCP/IP sockets. Internally, HPSS servers and clients attempt to set these buffers' sizes explicitly, but other utilities may not. Typically, the RecvSpace and the SendSpace are equal; however, this is not mandated. Setting either of these values in excess of the system maximum will result in a value less than or equal to the system maximum. There is no portable mechanism for determining the system maximum. Consequently, the specified values may be reduced until an acceptable value is obtained. This process involves a bit-shift operation (divide by two).

RFC1323. Controls whether large TCP window sizes are used. Usually turned on (1) for higher throughput networks (for example, SP/x switch or Gigabit Ethernet) and turned off (0) for lower throughput networks (such as 10/100 Mb Ethernet or FDDI). Large windows provide for increased performance over some networks, but may have a negative performance impact on others.

The **WriteSize** allows the size of the individual write requests to the TCP/IP connections to be configured. The default behavior (if no entry in the file matches a connection or if zero is entered for the value of this field) is that the size of the write request is the size of the data buffer. On some networks (for example, the SP/x switch), improved performance has been measured by using a smaller value (such as 32 KB) for the size of the individual writes to the network. If no entry is found that matches a network connection or the value specified is zero, HPSS will query an environment variable, **HPSS_TCP_WRITESIZE**, and use that value, if set and nonzero, for the write

size.

The **TcpNoDelay** option determines whether HPSS will enable or disable the algorithm that tries to improve performance from small network writes. This algorithm attempts to coalesce small writes to a TCP/IP connection so they can be sent in a single packet by delaying physical writes to the network. HPSS typically disables this algorithm so that delays are not experienced while sending Mover Protocol and parallel data transfer headers. However, if this causes a performance degradation on a specific network (for example, causes smaller than optimal packet sizes for large transfers), this can be disabled for data transfer connections.

Network Options Stanza Specific rules:

- The first matching entry found in the file will be used to determine the network options used for that connection.
- Multiple "Source Interface Name" SubStanzas may be included within the "Network Options" Stanza. A "Default" Source Interface Name SubStanza may be specified.
- The Source Interface Name SubStanza may specify one or more names (subject to the `HPSS.conf` line character limit, including the "= \{").



Do not include the quotes when specifying Default. Destination IP address must be specified in decimal dot notation. Multiple Sections may be included in any SubStanza. A "Default" Destination Interface Name Section may be specified.



Do not include the quotes when specifying Default. The NetMask must be specified in decimal dot IP address notation. All SubSections must be specified in every Section.

NOTE: Tuning is a "fine art" and may vary dramatically within any network configuration and may change with only very minor network configuration modifications. Values provided below are not necessarily "good" numbers.

Network Options Stanza example:

```
#  HPSS Network Options
Network Options = {
    ; Default Receive Size = 1MB
    ; Default Send Size = 1MB
    ; Default Write Size = 4MB
    # My Interface specification(s) using Interface Name
    # Notation
    my_host my_host.domain = {
        # Destination IP address in dot notation
        100.101.102.103 = {
            # The netmask to be applied to the Dest. IP address
            Netmask = 255.255.255.0
            # Use large IP Windows
            RFC1323 = 1
            # Socket Transmission Size.
            SendSpace = 1048576
```

```
            # Socket Receive Size.
            RecvSpace =  1MB
            # The overall buffer size to use for writing.
            WriteSize = 2MB
            # The TCP No Delay Flag is disabled
            TCPNoDelay = 0
        }
        # Default Destination - options to be used for destinations
        # NOT explicitly specified.
        Default = {
            NetMask = 255.255.255.0
            RFC1323 = 1
            SendSpace = 512KB
            RecvSpace =  512KB
            WriteSize = 256KB
            TCPNoDelay = 1
        }
    }
    # Values to be used for source hosts not explicitly specified
    Default  = {
        #  Destination IP address in dot notation
        200.201.202.203 = {
            NetMask = 255.255.255.0
            RFC1323 = 1
            SendSpace = 1048576
            RecvSpace =  1MB
            WriteSize = 2MB
            TCPNoDelay = 0
        }
        # Default Destination - options to be used for destinations
        # NOT explicitly specified.
        Default = {
            NetMask = 255.255.255.0
            RFC1323 = 1
            SendSpace = 256KB
            RecvSpace =  128KB
            WriteSize = 512KB
            TCPNoDelay = 0
        }
    }
}
```

D.5. PFTP Daemon Stanza

A large number of options are available for configuring the PFTP daemon and tuning its performance. These options were previously specified in the `ftpaccess` file or via command-line switches. These options have now been consolidated into the PFTP Daemon Stanza in the `HPSS.conf` file. The options are described below:

Table 37. PFTP Daemon Stanza description

Configuration type	Abbreviated description
Stanza (CMPD)	PFTP Daemon = \{ *Optional* Reserved_Stanza specifier on client machines *only*. Required Stanza on all HPSS PFTP server systems. Must be terminated with a matching "}"
SubStanza	Allow Core Files *Optional* SubStanza specifying that the system should save core files if the PFTP daemon crashes. By default, **xinetd** disables core files.
SubStanza	Core File Directory = <value> Value: **Pathname** Example: `Core File Directory = /var/hpss/adm/core/PFTP_Daemon` *Optional* SubStanza to specify the directory where the PFTP server should put core files
SubStanza	TA API Debug Level Values: 0 - 4 Example: `TA API Debug Level = 0` *Optional* SubStanza specifying the level of debugging for the transfer agent. A larger number provides more information.
SubStanza	TA API Debug Log Value: Debug log file path Example: `TA API Debug Log = /var/TA/logs/PFTPD_TA_API_debug_%N_%P.log` *Optional* SubStanza specifying the path to the transfer agent log file. A **%N** will be replaced by the number of agents, and a **%P** will be replaced by the PID of the particular agent.

Configuration type	Abbreviated description
SubStanza	SYSLOG Facility = <value> Value: DAEMON, LOCAL0 ... LOCAL7 Example: `SYSLOG Facility = LOCAL0` Replaces **-s<string>** option *Optional* SubStanza specifying the syslog facility for the HPSS PFTPD. The default syslog facility is **DAEMON** (reference: `/usr/include/sys/syslog.h`). Alternatives are **LOCAL0** - **LOCAL7**. Incorrect specification will default back to **DAEMON**. To make use of the alternates, modify `/etc/syslog.conf` to use the alternate facility. Note, the file specified in the `/etc/syslog.conf` must exist prior to initialization/refresh of the **syslogd**.
SubStanza	Use Foreign LDAP for Cross Realm Example: `Use Foreign LDAP for Cross Realm` *Optional* SubStanza specifying that LDAP lookups should use the LDAP server in the foreign realm. Few sites want this option.
SubStanza	Realms are Equivalent Value: off (default), on Example: `Realms are Equivalent = on` *Optional* SubStanza specifying that user in realm A is the entity as in realm B
SubStanza	FTP Base Directory = <value> Value: **Pathname** Example: `FTP Base Directory = /var/hpss` Replaces **-D<string>** option *Optional* SubStanza setting the **{FTPBaseDir}** path. Default: `/var/hpss`. This directory must contain several subdirectories including: `adm`, `bin`, `daemon`, and `etc`. Specific files and subdirectories are located in each of these subdirectories - `etc`: `ftpaccess`, [`ftpbanner`], and `ftpusers`. `adm`: [`daemon.syslog`], [`hpss_ftpd.log`], [`xferlog`]. `daemon`: `ftpd/ftp.pids-hpss_class`. [ ] implies optional others are required. `etc/passwd` is optional for FTP if **Use the KDC Registry** or **Use Extended Registry Attributes** is specified.

Configuration type	Abbreviated description
SubStanza	FTP Access File = <value> Value: **filename** Example: `FTP Access File = myftpaccess` Replaces **-F<string>** option *Optional* SubStanza setting the \{**FTP_FtpAccessFile**}. Default: **ftpaccess**. Located in the directory `{FTPBaseDir}/etc`.
SubStanza	Disable Slash Home Directory Example: `Disable Slash Home Directory` Replaces **-Z** option *Optional* SubStanza disabling use of "/" (forward slash) as the user's home directory. Normally, this should be active for security reasons.
SubStanza	Disable Access if no Home Directory Example: `Disable Access if No Home Directory` Replaces **-H** option *Optional* SubStanza disallowing login for users whose home directory does not exist or is not properly configured. The default behavior is to put the user in the "/" (forward slash) directory. Normally, this should be active for security reasons.
SubStanza	HPSS FTP Principal = <value> Value: Appropriate HPSS Principal Name Example: `HPSS FTP Principal = hpssftp` Replaces **-P** option and **hpss_option PRINC name** in `ftpaccess` *Optional* SubStanza specifying the HPSS principal representing **hpssftp** (HPSS only)
SubStanza	Disallow Passive Connections Example: `Disallow Passive Connections` *Optional* SubStanza disabling passive connections

Configuration type	Abbreviated description
SubStanza	Sleep for Debugger = <value> Example: `Sleep for Debugger = 5` Replaces **-z** option *Optional* SubStanza specifying the number of seconds for the HPSS PFTP daemon to sleep at initialization. Useful when attempting to attach to the daemon with the debugger. **NOTE**: leaving this active will cause significant degradation to the PFTP service.
SubStanza	Must Have Credentials Example: `Must Have Credentials` Replaces **-a** option *Optional* SubStanza mandating authentication with Kerberos credentials, disabling{Username}/{Password} authentication. This also disables the **user** command.
SubStanza	Allow CCC Command Example: `Allow CCC Command` *Optional* Kerberos option not relevant to the HPSS PFTP daemon
SubStanza	PFTP IO Buffer Size = <value> Example: `PFTP IO Buffer Size = 4MB` Replaces **-b<string>** option *Optional* SubStanza setting the preferred IO Buffer Size for the PFTP server
SubStanza	Debug Value = <value> Example: `Debug Value = 3` Replaces **-d** option(s) *Optional* SubStanza specifying the level of debugging desired (1 - 4). Used internally to determine the quantity and detail of syslog messages from the PFTP daemon.

Configuration type	Abbreviated description
SubStanza	Non-Parallel HostName = <value> Example: `Non-Parallel HostName = rahe.sandia.gov` Replaces **-h** option and **hpss_option HOSTname** in `ftpaccess` *Optional* SubStanza specifying the network interface to be used for data transferred between the PFTPD and the Movers when performing non-parallel transfers. Sets the **HPSS_API_HOSTNAME** environment variable for the Client API (HPSS only).
SubStanza	PFTP Debug Port = <value> Example: `PFTP Debug Port = 6666` Replaces **-p<port>** option *Optional* SubStanza specifying a port to be used by the HPSS PFTP daemon. Used only when initiating the daemon manually (rather than using **inetd/xinetd**). May be left on; it will not interfere with normal operations.
SubStanza	Default Time Out = <value> Example: `Default Time Out = 1500` Replaces **-t** option and **hpss_option DTO time** in `ftpaccess` *Optional* SubStanza specifying the default timeout in seconds
SubStanza	Default Umask = <value> Example: `Default Umask = 077` Replaces **-u** option and **hpss_option UMASK octal** in `ftpaccess` *Optional* SubStanza specifying the default umask in octal

Configuration type	Abbreviated description
SubStanza	Client API Verbose Value = <value> Example: `Client API Verbose Value = 1` Replaces **-v** option(s) *Optional* SubStanza specifying the level of HPSS Client API Logging to use (1 - 7). The Client API will perform logging specified by the **HPSS_DEBUG** environment variable in a file specified by the **HPSS_DEBUGPATH** environment variable. (Default name is `/var/hpss/ftp/adm/hpss_ftpd.log`.) The default value is **1** (HPSS only).
SubStanza	Disallow User Setting of COS Example: `Disallow User Setting of COS` Replaces **-C** option *Optional* SubStanza to disable the ability of clients to explicitly set the Class of Service for new files (via the **"site setcos"** command). Not recommended.
SubStanza	Maximum Time Out = <value> Value: Time in seconds Example: `Maximum Time Out = 86400` Replaces **-T** option and **hpss_option MTO time** in `ftpaccess` *Optional* SubStanza specifying the maximum timeout in seconds
SubStanza	Use Extended Registry Attributes Example: `Use Extended Registry Attributes` Replaces **-X** option *Optional* SubStanza specifying use of the LDAP registry for authentication (bypassing the **passwd** file) and use of the **HPSS.homedir** and **HPSS.gecos** Extended Registry Attributes (ERAs) for the user's home directory and accounting fields (if they are filled)

Configuration type	Abbreviated description
SubStanza	Print Performance Data Example: `Print Performance Data` *Optional* SubStanza specifying the printing of additional performance numbers (non-HPSS PFTP daemon only)
SubStanza	Number of Ports = <value> Values: 1 - 64 Example: `Number of Ports = 16` *Optional* SubStanza specifying the maximum stripe width allowed (non-HPSS PFTP daemon only)
SubStanza	PortRange = <value> Example: `PortRange = 10100-12100` *Optional* SubStanza specifying the port range to be used for the non-HPSS PFTP daemon which is necessary for parallel transfers. This is ignored for passive listings. The old format (ncacn_ip_tcp[10100-12100]) is still supported at present, but might be phased out in a future version of HPSS.
SubStanza	Socket Buffer Size = <value> Values: Viable Socket Sizes Example: `Socket Buffer Size = 1MB` *Optional* SubStanza specifying the socket buffer size (non-HPSS PFTP daemon only)
SubStanza	Set COS Based on Filesize Example: `Set COS Based on Filesize` *Optional* SubStanza specifying to set the COS from the FileSize Options table. Default: Ignore the COS in the table.

Configuration type	Abbreviated description
SubStanza (Compound)	FileSize Options = \{ Example: `FileSize Options = \{` *Optional* SubStanza specifier Must be terminated with a matching "}" See notes below
Section (Compound)	<value> = \{ Example: `1MB = \{` *Optional* Section specifier. Must be terminated with a matching "}" See notes below
SubSection	BlockSize = <value> Example: `BlockSize = 512KB` *Optional* SubSection specifying the size of data blocks to be used based on file size. (Has no meaning for the HPSS PFTP daemon.)
SubSection	StripeWidth = <value> Example: `StripeWidth = 0` *Optional* SubSection specifying the stripe width to be used based on file size. **0** (zero) means to use the Core Server Value (HPSS PFTP daemon) or use the default (non-HPSS PFTP daemon).
SubSection	COS = <value> Example: `COS = 2` *Optional* SubSection specifying the Class of Service to be used based on file size. **0** (zero) means to allow the Core Server to determine the optimal COS. (Has no meaning for the non-HPSS PFTD daemon.)

Configuration type	Abbreviated description
SubStanza	<nodename> Service Name = <canonicalname> Example: `sunrahe Service Name = sunrahe.sandia.gov` *Optional* SubStanza specifying the service name to be used by the PFTP daemon node when acquiring credentials. Needed when the servername in the keytab is different from that obtained by **gethostname()**. Use multiple entries when this file is common to multiple PFTP daemons. Useful particularly for clusters and systems having multiple names. One or the other of **host/servicename@realm** or **ftp/servicename@realm** must exist in the Kerberos KDC and in the `/etc/v5srvtab` for the PFTP server executing on the machine mymachine.ssm.com. (Non-HPSS PFTP daemon only.)
SubStanza	Use System Password Files = <value> Example: `Use System Password Files = TRUE` SubStanza specifying that the system password files (`/etc/passwd`, `/etc/group`, `/etc/shadow`) should be used. Should be specified explicitly. TRUE and FALSE are case-sensitive.
SubStanza	PFTP Password File = <value> Value: **Pathname/Filename** Example: `PFTP Password File = /var/hpss/etc/passwd` *Optional* SubStanza used to specify the file containing the user's password information. `/var/hpss/etc/passwd` is the default.
SubStanza	PFTP Shadow File = <value> Value: **Pathname/Filename** Example: `PFTP Shadow File = /var/hpss/etc/shadow` *Optional* SubStanza used to specify the file containing the user's protected password information. This should be specified if USERNAME/PASSWORD authentication is in effect.

Configuration type	Abbreviated description
SubStanza	PFTP Group File = <value> Value: **Pathname/Filename** Example: `PFTP Group File = /var/hpss/etc/groups` *Optional* SubStanza used to specify the file containing the group information for PFTP clients. Default is `/var/hpss/etc/group`.
SubStanza	Primary Authentication Type = <value> Values: **krb5** (default), **spkm** (Not supported?), **unix** Example: `Primary Authentication Type = krb5` *Optional* SubStanza used to specify the default authentication type
SubStanza	Primary Authenticator = <value> Values: <auth_type>:<auth_file> where <auth_type> = auth_keytab, auth_keyfile, auth_key, auth_password, auth_none Example: `Primary Authenticator = auth_keytab:/var/hpss/etc/hpss.keytab` *Optional* SubStanza used to specify the file containing the information to authenticate/authorize the **hpssftp** principal
SubStanza	Site Configuration File = <value> Values: **Pathname/Filename** Example: `Site Configuration File = /var/hpss/etc/site.conf` *Optional* SubStanza used to specify the site configuration file to use with the Primary Authentication Mechanism

Configuration type	Abbreviated description
SubStanza	Secure Authentication = <value> Values: **none**, **required**, **required_same**, **accepted** *Optional* Substanza used to specify the requirements for secure authentication. • **none** means it is not allowed • **required** means it is required, but the login provided in the USER command does not need to match. • **required_same** means that the USER command must match the authenticated name. • **accepted** means secure authentication is accepted but does not affect the USER/PASS cases. When the USER command does match the securely authenticated principal, the PASS command will be optional and ignored if given.
SubStanza	Mapfile Specifier = <value> Values: **Pathname/filename** Example: `Mapfile Specifier = /var/hpss/etc/MapfileName` *Optional* SubStanza used to specify a file containing username mappings. This file provides the ability to authenticate as one user and be authorized as another user (entity account). These files *must* be protected for security reasons. These files should be owned by *root* and readable and writable by *root* only.
SubSection	Default Authorization Mechanism = <value> Values: **LDAP**, **UNIX**, **DBAS** (not implemented) Example: `Default Authorization Mechanism = LDAP` *Optional* SubStanza used to specify the authorization mechanism desired. The PFTP daemon does authorization internally. If the HPSS system is configured to use LDAP and the PFTP server is configured to use UNIX, the end user will have to be in both authorization facilities. If the LDAP bind type is GSSAPI, the Primary Authorization Mechanism must be "krb5". If LDAP is specified for USER_PASS, a site method must also be specified. **DBAS** has *not* been implemented in PFTP or in HPSS.

Configuration type	Abbreviated description
SubSection	Use Site Auth Method = <value> Values: CryptoCard, KRB5KDC, SecurID, NIST Example: `Use Site Auth Method = CRYPTOCARD` *Optional* SubStanza used to specify a site-specific authentication mechanism to be used instead of the UserName/Password mechanism. This option requires a specific recompile of the hpss_pftpd(7) with site-specific modules linked in. **NOTE:** if this option is specified, the UserName/Password Mechanism will *not* use a standard password.
SubStanza	{Hostname} Service Name = {servicename} Example: `mymachine Service Name = fire.clearlake.ibm.com` *Optional* SubStanza used to specify alternate service names for the Kerberos service principals. The value after the equal sign is appended to either "host" or "ftp" to form a service by the name like: host/fire.clearlake.ibm.com@realm. This is very useful for computing clusters and multi-homed systems using a Kerberized PFTP server.

All SubStanzas are optional. If the optional SubStanza **FileSize Options** = **\{** is included, one or more <**value**> = **\{** Sections must be included with mandatory SubSections **BlockSize** = **...**, **StripeWidth** = **...**, and **COS** = **....**

Each <**value**> = **\{** Section defines the beginning of a range of file sizes to which its settings apply. That range begins with **value** and extends to the next larger **value** included in the **FileSize options** = **\{** SubStanza. In the example below, the settings in the **1MB** = **\{** Section below apply to files with sizes in the range [1MB, 2MB).

PFTP Daemon Stanza example:

```
PFTP Daemon = {
    # Allow the Daemon to take Core Dumps
    ; Allow Core Files
    # Directory to put core files in (Default = .)
    ; Core File Directory = /var/hpss/adm/core
    # The (optional) Agent Debug Level Stanza sets the
    # debug level (0-4, 0=no debug, 4=max debug) for the
    # Transfer Agent processes.  This overrides the command
    # line "-d" option for the agent, if specified in this file.
    TA API Debug Level = 0
    # The Debug Logfile contains debugging output. It can
    # be overridden by environment variable settings.
```

```
; TA API Debug Log = /var/TA/logs/PFTPD_TA_API_debug_%N_%P.log
#  Specify the SYSLOG facility to use for all syslog messages
#  except Authentication Messages.
#  Values: DAEMON, LOCAL<0-7>
#  Replaces -sstring option.    Default = DAEMON
SYSLOG Facility = LOCAL0
#  Use another cell's LDAP for cross realm authorization
;  Use Foreign LDAP for Cross Realm
#  User in realm A is the same entity as user in realm B
;  Realms are Equivalent = On
#  Specify the Base Directory for PFTP Files
;  FTP Base Directory = /var/hpss
#  Specify the name of the ftpaccess file
#  This becomes {BaseDir}/etc/{ftpaccess} based on the FTP Base Directory
;  FTP Access File = ftpaccess
#  Do NOT allow / to be Home Directory (HPSS Only)
Disable Slash Home Directory
#  Terminate session if Home Directory does NOT Exist (HPSS Only)
Disable Access if no Home Directory
#  What Principal to use for HPSS FTP (HPSS Only)
; HPSS FTP Principal = hpssftp
#  Disallow Passive Connections (HPSS Only)
; Disallow Passive Connections
#  Delay for xx seconds to allow dbx attach to the Daemon.
; Sleep for Debugger = 5
```

```
#  Allow the CCC Command to the GSS Daemon (non-HPSS Only)
#  This allow for the Control channel to be in clear text
; Allow CCC Command
#  Set the IO Buffer Size for the HPSS PFTP Daemon
; PFTP IO Buffer Size = 1MB
#  Specify the Level of Debugging Desired.
; Debug Value = 1
#  For non-Parallel Transfers, specify the Interface (by Name)
#  to use between the PFTP Daemon and the Movers (HPSS Only)
#  Replaces -h option and "hpss_option HOST name" in ftpaccess
; Non-Parallel HostName = rahe.sandia.gov
#  Specify the Port to be used for Manual PFTP Daemon Startup
; PFTP Debug Port = 6666
#  Specification in seconds for the Default Timeout
; Default Time Out = 1500
#  Specify (in octal) the Default umask
; Default Umask = 077
#  Specification of the Level of HPSS Client API logging to use ( 0 - 7 )
; Client API Verbose Value = 0
#  Do NOT allow the user to specify Classes of Service (HPSS Only)
; Disallow User Setting of COS
#  Specification in seconds for the Maximum Timeout
; Maximum Time Out = 86400
#  Use the Extended Registry Attributes if they Exist (HPSS.conf)
```

```
; Use Extended Registry Attributes
#  Print additional Performance Numbers (non-HPSS PFTP Daemon Only)
; Print Performance Data
#  Specify the Maximum Stripe Width Allowed (non-HPSS PFTP Daemon Only)
; Number of Ports = 16
#  The Port Range to be used for the non-HPSS Parallel FTP Daemon
#  which is necessary for Parallel Transfers (non-HPSS PFTP Daemon Only)
; PortRange = 10100-12100
#  The Socket Buffer Size (non-HPSS PFTP Daemon Only)
; Socket Buffer Size = 1MB
#  Uncomment next line to use the COS from the FileSize Options
#  The default is to ignore the COS specification in the Table.
; Set COS Based on Filesize
#  Specify Blocksizes, StripeWidths, and COSs to use based on file size
#  COS has no meaning to the Non-HPSS PFTP Daemon
#  COS = 0 means allow the BitFile Server to determine the optimal COS
#  BlockSize has no meaning to the HPSS PFTP Daemon Only
#  StripeWidth = 0 means use the Bitfile Server Value (HPSS PFTP Daemon)
#      or use the default (Non-HPSS PFTP Daemon)
```

```
; FileSize Options = {
#  Files greater than or equal to this value and less than
#  any other value in the table use these Settings
#  e.g., 2MB <= filesize < 10MB
; 1MB = {
    ; BlockSize = 512KB
    ; StripeWidth = 0
    ; COS = 2
; }
; 2MB = {
    ; BlockSize = 2MB
    ; StripeWidth = 4
    ; COS = 0
; }
; 10MB = {
    ; BlockSize = 2MB
    ; StripeWidth= 0
    ; COS = 0
; }
; 100MB = {
    ; BlockSize = 4MB
    ; StripeWidth= 0
    ; COS = 0
; }
; 1GB = {
    ; BlockSize = 8MB
    ; StripeWidth= 0
    ; COS = 0
; }
; }
```

```
# Use the System Password file routines (TRUE or FALSE)
# The Default for PFTP is FALSE (Case Sensitive!)
Use System Password Files = FALSE

#  Path and Name for the PFTP Password File
PFTP Password File = /var/hpss/etc/passwd

#  Path and Name for the PFTP Shadow Password File
# NOTE: PFTP does not currently use the value. It is used ONLY to
# change how the password is looked up! If the site is using
# /etc/passwd and the system running the PFTP Daemon utilizes
# some form of "Shadow" password file to authenticate PFTP users,
# this should be uncommented.
# Do NOT remove the part after the "=" sign.
;  PFTP Shadow File = /etc/security/passwd

#  Path and Name for the PFTP Group File
;  PFTP Group File = /etc/group

# Primary Authentication Type for the FTP Daemon
# This mechanism will be used to authenticate "hpssftp" with HPSS
# using the PFTP Daemon.
# Default: krb5 - Options: krb5, spkm(Not supported?), unix
; Primary Authentication Mechanism = krb5
```

```
# Primary Authenticator for the FTP Daemon
# This mechanism will be used to authenticate "hpssftp" with HPSS
# using the PFTP Daemon.
# Format: <auth_type>:<auth_file>
#         where <auth_type> = auth_keytab, auth_keyfile, auth_key,
#                             auth_passwd, auth_none
# Default: auth_keytab:/var/hpss/etc/hpss.keytab
; Primary Authenticator = auth_keytab:/var/hpss/etc/hpss.keytab

# Mapfile Specifier specifies the type and required information
#       for Mapping one user name to another.  The types include
#       "FILE:", "LDAP:", and "DBAS:" The default type is "FILE:"
#       For "FILE:" specify the path and name of the file after the ":"
#       "LDAP" and "DBAS" are NOT currently supported.
#
; Mapfile Specifier = FILE:/var/hpss/etc/KRB2UnixMapfile

#  Keytab Hostname Mapping Section
#  Syntax:
#      machinename Service Name = canonicalname
#  "machinename" is the name returned by a call to gethostname()
#  in the PFTP Daemon.
#  "canonicalname" is the machine name associated with the Kerberos
#  service - usually the fully qualified name as generated by the
```

```
    #  PFTP Client
    #  Specify "{machinename} Service Name" to set the service name to be used
    #  for the PFTP Daemon machine when acquiring creds.  This is needed
    #  when the servername; e.g., host/machinename@realm is different
    #  between the keytab file and the host/machinename obtained where
    #  the machinename is obtained by gethostname() (Non-HPSS PFTP Daemon)
    #  Specify nultiple "machinename Service Name" entries if this
    #  file is common to multiple PFTP Daemon servers.
    ; rahe.sandia.gov Service Name = rahe.sandia.gov
    ; sunrahe Service Name = sunrahe.sandia.gov
}
```

D.6. Transfer Agent Stanza

A large number of options are available for configuring the transfer agent and tuning its performance.

Table 38. Transfer Agent Stanza description

Configuration type	Abbreviated description
Stanza (CMPD)	Transfer Agent = \{ *Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza	Agent Debug Level = <value> Value: 0 - 4 Example: `Agent Debug Level = 1` *Optional* Sets the debug level (where 0=no debug, 4=max debug) for transfer agent processes
SubStanza	Debug Log File = <value> Value: **Pathname/filename** Example: `Debug Logfile = /var/TA/logs/agent_debug_%N_%P.log` *Optional* Specifies the debugging output file. It can be overridden by environment variable settings.

Configuration type	Abbreviated description
SubStanza	Enable Core Dump = <value> Value: **YES/NO** Example: `Enable Core Dump = YES` *Optional* Determines whether the agent will enable full core dumps
SubStanza	Working Directory = <value> Value: **Pathname** Example: `Working Directory = /var/TA/cores` *Optional* The absolute pathname of the directory to which the agent will "cd" upon startup, so that core and other important files will be in a known location. The default, if not specified, is the $TMPDIR environment variable setting, or "/tmp" as an absolute fallback.
SubStanza	Nodeset File = <value> Value: **Pathname/filename** Example: `Nodeset File = /usr/local/etc/nodeset.conf` *Optional* File containing named sets of nodes that can be referred to via the "SET:setname" notation
SubStanza	Node Affinity File = <value> Value: **Pathname/filename** Example: `Node Affinity File = /usr/local/etc/node_affinity.conf` *Optional* File used to specify groups of nodes that are able to communicate in a network whose topology does not support full interconnection
SubStanza	Shared FS File = <value> Value: **Pathname/filename** Example: `Shared FS File = /usr/local/etc/shared_fs.conf` *Optional* File used to specify the list of shared file system mount points and associated nodes or NodeSets

Configuration type	Abbreviated description
SubStanza	Agent File = <value> Value: **Pathname/filename** Example: `Agent File = /usr/local/etc/agent.conf` *Optional* File containing the list of client hosts and the list of nodes that can be used as agents from each client
SubStanza	Disabled Node File = <value> Value: **Pathname/filename** Example: `Disabled Node File = /usr/local/etc/disabled_node.conf` *Optional* File used to disable selection of nodes that are temporarily unavailable. It overrides entries in the Agent File.
SubStanza	Audit Logfile = <value> Value: **Pathname/filename** Example: `Audit Logfile = /var/hpss/log/PMTA_AuditLog` *Optional* File used to specify the location of the transfer agent audit log
SubStanza	Debug Logfile = <value> Value: **Pathname/filename** Example: `Debug Logfile = /var/TA/log/pmta_Audit_%N_%P.log` *Optional* Contains an audit trail of all PMTA activity
SubStanza	Agent Auth Mechanism = <value> Value: **none (default), ident, kerberos** Example: `Agent Auth Mechanism = none` *Optional* The authentication mechanism used by the agent to verify its parent's identity

Configuration type	Abbreviated description
SubStanza	Agent Authenticator Type = <value> Value: **none (ident), token (future), kerberos (future)** Example: `Agent Authenticator Type = none` *Optional* The mechanism-specific type of authenticator used for verifying the parent's identity
SubStanza	Agent Authenticator File = <value> Value: **Pathname/Filename** Example: `Agent Authenticator File = /usr/local/etc/ident_hosts.conf` *Optional* The absolute pathname of the authentication-specific authenticator file. For "ident", this file contains a list of trusted host patterns that are allowed to launch the transfer agents
SubStanza	SYSLOG Facility = <value> Values: **off**, **LOCAL0** ... **LOCAL7** Example: `SYSLOG Facility = LOCAL0` *Optional* Controls logging to syslog
SubStanza	Allow Uid Mapping = <value> Values: **YES, NO (default)** Example: `Allow Uid Mapping = YES` *Optional* Specifies whether the same user can have different UIDs on different machines within the site
SubStanza	Uid Mapfile = <value> Value: **Pathname/filename** Example: `Uid Mapfile = /usr/local/etc/uid_mapfile` *Optional* The name of the mapping file when "Allow Uid Mapping" is set to "YES"

Configuration type	Abbreviated description
SubStanza	Allow Gid Mapping = <value> Values: **YES, NO (default)** Example: `Allow Gid Mapping = YES` *Optional* Specifies whether the same group can have different GIDs on different machines within the site
SubStanza	Gid Mapfile = <value> Value: **Pathname/filename** Example: `Gid Mapfile = /usr/local/etc/gid_mapfile` *Optional* The name of the mapping file when "Allow Gid Mapping" is set to "YES"
SubStanza	Umask = <value> Value: **Octal 000 -** 777 Example: `Umask = 002` *Optional* Umask value that the transfer agent should set when it starts up
SubStanza	Connections per NIC = <value> Value: **1 (default) - 8** Example: `Connections per NIC = 4` *Optional* The number of data connections that should be opened for each network interface. This should be set to the same value on all agent nodes.
SubStanza	Max Local IO Count = <value> Value: **1 - 8 (default = 4)** Example: `Max Local IO Count = 4` *Optional* The maximum number of concurrent local file I/O ops

Configuration type	Abbreviated description
SubStanza	File Create Mode = <value> Value: **Octal 000 - 777** Example: `File Create Mode = 640` *Optional* The permissions that should be specified when the transfer agent creates files
SubStanza	Stripe Size = <value> Value: **128KB - 256MB** Example: `Stripe Size = 16MB` *Optional* The default number of bytes that each agent should write within each stripe of data. The total stripe length is (Stripe Size × number of agents). The value may contain the K, KB, M, or MB suffix. This value may be overridden by the application.

All Stanza and Sub Components are optional.

Transfer Agent Stanza example:

```
# Parallel Multinode Transfer Agent (PMTA) Section
Transfer Agent = {
        # The (optional) Agent Debug Level Stanza sets the
        # debug level (0-4, 0=no debug, 4=max debug) for the
        # Transfer Agent processes.  This overrides the command
        # line "-d" option for the agent, if specified in this file.
        Agent Debug Level = 0

        # The Debug Logfile contains debugging output. It can
        # be overridden by environment variable settings.
        ; Debug Logfile = /var/TA/logs/agent_debug_%N_%P.log

        # Enable Core Dump - this option determines whether the agent will
        # enable full core dumps. Allowable values for this option are YES and
        # NO.  The Default is YES
        Enable Core Dump = yes

        # The optional "Working Directory" setting is the absolute
        # pathname of the directory to which the agent will "cd" upon
        # startup, so that core files, etc. will be in a known location.
        # The default, if not specified, is the $TMPDIR environment
        #  variable setting, or "/tmp" as an absolute fallback.
        Working Directory = /var/TA/cores
```

```
# The (optional) NodeSet File contains named sets of
# Nodes that can be referred to via the "SET:setname"
# notation.
Nodeset File = /usr/local/etc/nodeset.conf

# The (optional) Node Affinity file is used to specify
# groups of nodes are able to communicate in a
# network whose topology does not support full interconnection
Node Affinity File = /usr/local/etc/node_affinity.conf

# The Shared Filesystem file is used to specify the
# list of shared filesystem mount points and associated
# nodes or NodeSets
Shared FS File = /usr/local/etc/shared_fs.conf

# The Agent file contains the list of client hosts and
# the list of nodes that can be used as Agents from each client
Agent File = /usr/local/etc/agent.conf

# The Disabled Node file is used to disable selection of
# nodes that are temporarily unavailable. It overrides
# entries in the Agent file
Disabled Node File = /usr/local/etc/disabled_node.conf

# The Audit Logfile contains an audit trail of all PMTA activity
; Audit Logfile = /var/TA/logs/pmta_Audit_%N_%P.log

# "Agent Auth Mechanism", if specified, is the authentication
# mechanism used by the Agent to verify its parent's identity.
# Valid settings are:
#   none   (default)
#   ident
#   kerberos
; Agent Auth Mechanism = none
```

```
# "Agent Authenticator Type" is the mechanism-specific type of
# authenticator used for verifying the parent's identity
# Legal settings are:
#   none  [for ident authentication]
#   token [reserved for future use]
#   kerberos [kerberos authentication - TBD]
; Agent Authenticator Type = none

# The "Agent Authenticator File", if specified, is the absolute
# pathname of the authentication-specific authenticator file.
# For "ident", this file contains a list of trusted host patterns
# that are allowed to launch the Transfer Agents
; Agent Authenticator File = /usr/local/etc/ident_hosts.conf

# The SYSLOG Facility parameter controls logging to syslog. Legal
```

```
# values are "off" or "LOCALn" where n=0-7.
SYSLOG Facility = off

# The optional "Allow Uid Mapping" setting defines whether the same user
# can have different UIDs on different machines within the site.
# The default value is NO
Allow Uid Mapping = NO

# The optional "Uid Mapfile" setting is the name of the
# mapping file when "Allow Uid Mapping" is set to "YES"
; Uid Mapfile = /usr/local/etc/uid_mapfile

# The optional "Allow Gid Mapping" setting defines whether the same group
# can have different GIDs on different machines within the site.
# The default value is NO
Allow Gid Mapping = NO

# The optional "Gid Mapfile" setting is the name of the mapping
# file used when "Allow Gid Mapping" is set to "YES"
; Gid Mapfile = /usr/local/etc/gid_mapfile

# The optional "Umask" setting is the value that the Transfer Agent
# should set when it starts up.  It is specified as a 3-digit
# octal value.
; Umask = 002

# The optional "Connections Per NIC" setting specifies the number of
# data connections that should be opened for each network interface.
# This should be set to the same value on all agent nodes.
# The default is one, and the max allowed is 8.
; Connections Per NIC = 4

# The max number of concurrent local file I/O ops
# 0 < value <= 8, Default = 4
; Max Local IO Count = 4

# The optional "File Create Mode" setting specifies the permissions
# that should be specified when the Transfer Agent creates files.
# It is specified as an octal value
File Create Mode = 640

# The optional "Stripe Size" setting is used to specify the default
# number of bytes that each Agent should write within each stripe of
# data.  The total stripe length is (Stripe Size * number of Agents).
# The value may contain K, KB, M, or MB suffix, and must be within
# the range (128K - 256MB)
# This value may be overridden by the application.
Stripe Size = 16MB

# The optional "Size To Agent Count" SubStanza is used to define the
# number of agents to be used for a transfer, based upon the transfer
```

```
        # size
        # Entry format for each interval is:
        #       AgentCount      MinSize MaxSize
        Size To Agent Count = {
                 0                0     700MB
                 1              700MB     2GB
                 2                2GB    40GB
                 4               40GB   200GB
                 8              200GB   500GB
                16              500GB
        # Note: omitted MaxSize value means 2^64-1
        } # end of Size To Agent Count Section
}
#  end of Transfer Agent Section
```

D.7. Stanzas reserved for future use

The following Stanza names (specifiers) are reserved for future implementation in HPSS and should not be used by application developers.

- Local File Path
- PSI

Chapter 31. HPSS Environment Variables

31.1. HPSS Environment Variables

The HPSS environment variables are defined in `/opt/hpss/include/hpss_env_defs.h`. These environment variables can be overridden in `/var/hpss/etc/env.conf` or in the local environment. Note that the default value for each environment variable is shown in parenthesis.

31.2. HPSS Misc Environment

Misc environment variables

HPSS_ROOT (`HPSS_ROOT`)

Root pathname for HPSS Unix top level

HPSS_HOST (`"%H"`)

Machine host name

HPSS_NODE_TYPE (`NULL`)

Node type of current machine

HPSS_PATH_INSTALL (`HPSS_PATH_INSTALL`)

Pathname for HPSS installation

HPSS_PATH_BIN (`"${HPSS_PATH_INSTALL}/bin"`)

Pathname for HPSS bin directory

HPSS_PATH_SLASH_BIN (`"/bin"`)

Pathname for bin directory

HPSS_PATH_SLASH_ETC (`"/etc"`)

Pathname for etc directory

HPSS_PATH_USR_BIN (`"/usr/bin"`)

Pathname for usr bin

HPSS_PATH_USR_SBIN (`"/usr/sbin"`)

Pathname for super user bin

HPSS_PATH_VAR (`"/var/hpss"`)

Pathname for HPSS var directory

HPSS_USER (`"hpss"`)

HPSS user name

HPSS_USERROOT (`"root"`)

Root user ID

HPSS_PATH_DB_INSTALL (`HPSS_PATH_DB_INSTALL`)

Pathname for DB bin

HPSS_SYSTEM (`"%S"`)

HPSS system name

HPSS_SYSTEM_VERSION (`"%V"`)

HPSS version

HPSS_HOST_FULL_NAME (`"%L"`)

Long hostname

HPSS_NET_FAMILY (`"ipv4_only"`)

Default network protocol to be used

HPSS_BATCH_TIMEOUT (`"120"`)

Timeout for MPS batch migration

HPSS_SOCK_KEEPIDLE_SECS (`"7200"`)

Socket keep alive seconds

HPSS_SOCK_KEEPIDLE_CNT (`"9"`)

Socket keep alive count

HPSS_SOCK_KEEPIDLE_INTVL (`"75"`)

Socket keep alive probe interval

31.3. HPSS Server Group Names

Environment variables related to HPSS server group names

HPSS_GRP_NAME (`"hpss"`)

HPSS group name

HPSS_GRP_NAME_SERVER (`"hpsssrvr"`)

HPSS server group name

31.4. HPSS Server Principal Names

Environment variables related to HPSS Server Principal Names

HPSS_PRINCIPAL_CORE (`"hpsscore"`)

Principal name for CORE server

HPSS_PRINCIPAL_DMG (`"hpssdmg"`)

Principal name for GHI DMAPI user

HPSS_PRINCIPAL_FTPD ("hpssftp")
Principal name for FTP Daemon

HPSS_PRINCIPAL_GK ("hpssgk")
Principal name for Gatekeeper Server

HPSS_PRINCIPAL_HPSSD ("hpsssd")
Principal name for Startup Daemon

HPSS_PRINCIPAL_FS ("hpssfs")
Principal name for HPSS FS

HPSS_PRINCIPAL_MPS ("hpssmps")
Principal name for Migration/PUrge Server

HPSS_PRINCIPAL_MVR ("hpssmvr")
Principal name for Mover

HPSS_PRINCIPAL_PVL ("hpsspvl")
Principal name for PVL

HPSS_PRINCIPAL_PVR ("hpsspvr")
Principal name for PVR

HPSS_PRINCIPAL_RAIT ("hpssrait")
Principal name for RAIT

HPSS_PRINCIPAL_SSM ("hpssssm")
Principal name for SSM

HPSS_PRINCIPAL_ADM_USER ("${HPSS_USER}")
Principal name for primary administrator

31.5. HPSS Server Principal UIDs

Environment variables related to HPSS Server Principal UIDs

HPSS_PRINCIPAL_CORE_UID ("301")
UID for CORE

HPSS_PRINCIPAL_FTPD_UID ("302")
UID for FTPD

HPSS_PRINCIPAL_GK_UID ("303")
UID for Gatekeeper

HPSS_PRINCIPAL_HPSSD_UID ("304")
UID for Startup Daemon

HPSS_PRINCIPAL_FS_UID ("306")

UID for HPSSFS

HPSS_PRINCIPAL_MPS_UID ("307")

UID for Migration/Purge

HPSS_PRINCIPAL_MVR_UID ("308")

UID for Mover

HPSS_PRINCIPAL_PVL_UID ("309")

UID for PVL

HPSS_PRINCIPAL_PVR_UID ("310")

UID for PVR

HPSS_PRINCIPAL_RAIT_UID ("311")

UID for RAIT

HPSS_PRINCIPAL_SSM_UID ("312")

UID for SSM

31.6. HPSS Server Executable Names

Environment variables related to HPSS Server Executable Names

HPSS_EXEC_ACCT ("${HPSS_PATH_BIN}/hpss_acct")

Executable path for Accounting

HPSS_EXEC_CORE ("${HPSS_PATH_BIN}/hpss_core")

Executable path for Core Server

HPSS_EXEC_FTPD ("${HPSS_PATH_BIN}/hpss_pftpd")

Executable path for pftpd

HPSS_EXEC_GK ("${HPSS_PATH_BIN}/hpss_gk")

Executable path for Gatekeeper

HPSS_EXEC_HPSSD ("${HPSS_PATH_BIN}/hpss_sd")

Executable path for Startup Daemon

HPSS_EXEC_MPS ("${HPSS_PATH_BIN}/hpss_mps")

Executable path for Migration / Purge Server

HPSS_EXEC_MVR ("${HPSS_PATH_BIN}/hpss_mvr")

Executable path for Mover

HPSS_EXEC_MVR_TCP ("${HPSS_PATH_BIN}/hpss_mvr_tcp")

Executable path for Mover TCP process

HPSS_EXEC_PVL ("${HPSS_PATH_BIN}/hpss_pvl")

Executable path for PVL

HPSS_EXEC_PVR_OPER ("${HPSS_PATH_BIN}/hpss_pvr_operator")

Executable path for Operator PVR

HPSS_EXEC_PVR_STK ("${HPSS_PATH_BIN}/hpss_pvr_stk")

Executable path for STK PVR

HPSS_EXEC_PVR_AML ("${HPSS_PATH_BIN}/hpss_pvr_aml")

Executable path for AML PVR

HPSS_EXEC_PVR_SCSI ("${HPSS_PATH_BIN}/hpss_pvr_scsi")

Executable path for SCSI PVR

HPSS_EXEC_RAIT ("${HPSS_PATH_BIN}/hpss_rait_engine")

Executable path for RAIT Engine

HPSS_EXEC_RAIT_TCP ("${HPSS_PATH_BIN}/hpss_rait_engine_tcp")

Executable path for RAIT Engine TCP process

HPSS_EXEC_SSMSM ("${HPSS_PATH_BIN}/hpss_ssmsm")

Executable path for System Manager

31.7. SSM Utility locations

Environment variables related to SSM utilities

HPSS_EXEC_DELOG ("${HPSS_PATH_BIN}/hpss_delog")

Pathname for the delog utility

HPSS_EXEC_RECLAIM ("${HPSS_PATH_BIN}/reclaim")

Pathname for the reclaim utility

HPSS_EXEC_REPACK ("${HPSS_PATH_BIN}/repack")

Pathname for the repack utility

31.8. Logging unix files

Environment variables related to logging unix files

HPSS_PATH_LOG ("${HPSS_PATH_VAR}/log")

Pathname for logging files

HPSS_LOG_MSG_LEN ("65535")

Split syslog messages which exceed this length

31.9. Accouting Unix files

Environment variables related to Accounting Unix files

HPSS_PATH_ACCT ("${HPSS_PATH_VAR}/acct")

Unix path name for accounting files

HPSS_UNIX_ACCT_CHECKPOINT ("${HPSS_PATH_ACCT}/acct_checkpoint")

Unix path name for checkpoint files

HPSS_UNIX_ACCT_REPORT ("${HPSS_PATH_ACCT}/acct_report")

Unix path name for accounting report

HPSS_UNIX_ACCT_COMMENTARY ("${HPSS_PATH_ACCT}/acct_commentary")

Unix path name for accounting commentary

31.10. MPS Unix files

Environment variables related to MPS unix files

HPSS_PATH_MPS ("${HPSS_PATH_VAR}/mps")

Unix path name for MPS files

HPSS_UNIX_MPS_REPORT ("${HPSS_PATH_MPS}/mps_report")

Unix path name for MPS reports

31.11. Gatekeeper environment variables

Environment variables related to the gatekeeper

HPSS_PATH_GK ("${HPSS_PATH_VAR}/gk")

Unix path name for Gatekeeping

HPSS_UNIX_GK_SITE_POLICY ("${HPSS_PATH_GK}/gksitepolicy")

Unix path name for Gatekeeping site policy

31.12. Database environment variables

Environment variables related to database

HPSS_DB_INSTANCE_OWNER ("hpssdb")

HPSS database instance owner

HPSS_GLOBAL_DB_NAME ("cfg")

Default name for the global database

HPSS_SUBSYS_DB_NAME ("subsys")

Default name for the subsys database

HPSS_SUBSYS_ID ("1")

Default subsystem ID for HPSS command line tools

HPSS_MM_SCHEMA_NAME ("HPSS")

HPSS schema name

HPSS_MM_STMT_CACHE ("75")

Sets the length of the MMLIB statement cache

HPSS_SERVER_DB_GROUP ("hpsssrvr")

Database auth group identity used by HPSS servers

HPSS_SERVER_DB_KEYTAB ("${HPSS_PATH_ETC}/mm.keytab")

Database connection keytab used by HPSS servers

HPSS_DISABLE_RTLD_DEEPBIND ("1")

Allow HPSS to run address-sanitizer (ASAN) binaries against Db2 11.5.9+

31.13. System environment variables

General system environment variables

HPSS_HOST_TMP ("%H")

The short form of the hostname

HPSS_DESC_CORE ("Core Server")

The default descriptive name for the core server

HPSS_DESC_FTPD ("FTP Daemon")

The default descriptive name for the FTP daemon

HPSS_DESC_GK ("Gatekeeper")

The default descriptive name for the gatekeeper

HPSS_DESC_HPSSD ("Startup Daemon (${HPSS_HOST_TMP})")

The default descriptive name for the startup daemon

HPSS_DESC_MPS ("Migration/Purge Server")

The default descripive name for the migration/purge server

HPSS_DESC_MVR ("Mover (${HPSS_HOST_TMP})")

The default descriptive name for the mover

HPSS_DESC_PVL ("PVL")

The default descriptive name for the PVL

HPSS_DESC_PVR_OPER ("Operator PVR")

The default descriptive name for the Operator PVR

HPSS_DESC_PVR_STK ("STK PVR")

The default descriptive name for the STK PVR

HPSS_DESC_PVR_AML ("AML PVR")

The default descriptive name for the AML PVR

HPSS_DESC_PVR_SCSI ("SCSI PVR")

The default descriptive name for the SCSI PVR

HPSS_DESC_RAIT ("RAIT Engine (${HPSS_HOST_TMP})")

The default descriptive name for the RAIT Engine

HPSS_DESC_SSMSM ("SSM System Manager")

The default descriptive name for the SSM System Manager

31.14. SSM Specific

Environment variables related to SSM

HPSS_PATH_SSM ("${HPSS_PATH_VAR}/ssm")

Path name for data server files

HPSS_SSM_ALARMS (NULL)

File to store SSM Alarms & Events

HPSS_SSM_ALARMS_DISPLAY ("10000")

Number of SSM Alarms / Events to display and store

HPSS_SSM_MAX_IDLE_SYSLOG ("60")

Max number of minutes syslog file can be idle before getting concerned

HPSS_SSM_ALARMS_GET ("500")

Number of SSM Alarms / Events to get at one time

HPSS_SSM_COUNTRY ("US")

Country for Java internationalization

HPSS_SSM_LANGUAGE ("en")

Language for Java internationalization

HPSS_SSM_SERVER_LISTEN_PORT ("0")

Port the SM will listen on for client RPCs. If 0, port will be chosen by the portmapper.

HPSS_SSM_TIMING_DEBUG ("0")

Turns on/off SSM timing debug logging

HPSS_SSM_CLI_PRINCIPAL ("${HPSS_PRINCIPAL_SSM}")

SSM API administrator account

HPSS_SSM_CLI_AUTHENTICATOR ("${HPSS_PRIMARY_AUTHENTICATOR}")

Authenticator for HPSS_SSM_CLI_PRINCIPAL

HPSS_HELP_FILES_PATH ("${HPSS_PATH_INSTALL}/ssmhelp")

HPSS Help files install path

HPSS_HELP_URL_TYPE ("file:")

HPSS help files URL

HPSSGUI_SM_HOST_NAME ("${HPSS_HOST}")

Host that SM is on

HPSSADM_SM_HOST_NAME ("${HPSS_HOST}")

Host that SM is on

HPSSGUI_SM_PORT_NUM ("536870913:1")

Port that SM is on

HPSSADM_SM_PORT_NUM ("536870913:1")

Port that SM is on

HPSSGUI_RPC_PROT_LEVEL ("${HPSS_RPC_PROT_LEVEL}")

RPC protection level for SM communication

HPSSADM_RPC_PROT_LEVEL ("${HPSS_RPC_PROT_LEVEL}")

RPC protection level for SM communication

HPSSSSM_UI_WAIT_TIME ("${SSM_UI_WAIT_TIME}")

Time that GUI will wait at the SM for updates

HPSSSSM_UI_MO_RATE ("${SSM_UI_MO_RATE}")

Time the GUI will wait between managed object (MO) updates

HPSSSSM_UI_LIST_RATE ("${SSM_UI_LIST_RATE}")

Time the GUI will wait between list update tries

HPSSSSM_UI_ALARM_RATE ("${SSM_UI_ALARM_RATE}")

Time the GUI will wait between alarm update tries

HPSS_SSM_SEC_MECH (NULL)

Security mechanism for SM communication

HPSSGUI_USER_CFG_PATH ("${HOME}/hpss-ssm-prefs")

Directory which holds GUI config files

HPSSADM_USER_CFG_PATH ("${HOME}/hpss-ssm-prefs")

Directory which holds ADM config files

HPSS_SSMUSER_JAVA_POLICY ("${HPSS_PATH_VAR}/ssm/java.policy.ssmuser")

Java policy file for SSM GUI and hpssadm

JAVA_CLS1 ("${HPSS_ROOT}/bin/hpss.jar")

Location of HPSS SSM Java classes

HPSS_SSM_CLASSPATH ("${JAVA_CLS1}")

Runtime search path for Java classes

HPSS_SM_SRV_CONNECT_FAIL_COUNT ("3")

Number of connection failures to a server before the HPSS_SM_SRV_CONNECT_INTERVAL_MAX interval takes effect.

HPSS_SM_SRV_CONNECT_INTERVAL_MIN ("20")

Interval between attempting server connections when HPSS_SM_SERVER_CONNECT_FAIL_COUNT has not yet been reached (seconds)

HPSS_SM_SRV_CONNECT_INTERVAL_MAX ("60")

Interval between server connections when HPSS_SM_SERVER_CONNECT_FAIL_COUNT has been reached without a successful connection (seconds)

HPSS_SM_SRV_LIST_UPDATE_INTERVAL ("5")

Frequency at which the SM updates the server list (seconds)

HPSS_SM_SRV_MONITOR_THREADS ("10")

Number of threads created to monitor server connections

HPSS_SM_SRV_QUEUE_SIZE ("0")

Request Queue Size used by the System Manager server interface - default of 0 means that there will be 20 slots in the server interface request queue to be used when the server interface threadpool is completely full. The queue is used to hold RPC requests from servers until a thread is available to process the request. Note that if the request queue has any entries in it it means that all the threads in the server thread pool are busy and the SM response will be degraded. If this happens then it would be good to increase the number of threads available to the server interface using the HPSS_SM_SRV_TPOOL_SIZE variable. Increasing the size of the queue will not help with performance.

HPSS_SM_SRV_TPOOL_SIZE ("100")

Thread Pool Size used by the System Manager server interface. If the Thread Pool is exhausted then server RPC requests will be queued in the server RPC Request Queue to wait for a thread to become available. When the thread pool is exhausted SM performance may be degraded. Increase this value if that is the case. Typically 1 thread per HPSS server should be adequate. But a few extra wouldn't hurt.

HPSS_SM_SRV_MAX_CONNECTIONS ("50")

Number of HPSS server connections to maintain at once. If this number of connections is exceeded, then old connections will be close to maintain this number of connections

31.15. API Specific

Environment variables related to API

HPSS_API_HOSTNAME ("${HPSS_HOST}")

Used to control the network interface selected for data transfers.

HPSS_API_DEBUG_PATH ("stdout")

Used to direct debug output messages

HPSS_API_MAX_CONN ("0")

Defines the number of connections that are supported by the Client API within a single client process (0 == unlimited)

HPSS_API_MAX_OPEN ("4096")

Defines the maximum number of open files

HPSS_API_DEBUG ("0")

Used to enable debug messages

HPSS_API_RETRIES ("4")

Used to control the number of retries when operations fail with a "retryable" return code

HPSS_API_BUSY_DELAY ("15")

Used to control the number of seconds to delay between retry attempts.

HPSS_API_BUSY_RETRIES ("3")

Used to control the number of retries to be performed when a request fails because the Core Server does not currently have an available thread to handle the request

HPSS_API_TOTAL_DELAY ("0")

Used to control the number of seconds to continue retrying a request

HPSS_API_LIMITED_RETRIES ("1")

Used to control the number of retry attempts before a limited retry error operation fails

HPSS_API_DMAP_WRITE_UPDATES ("20")

Used to control the frequency of the cache invalidates that are issued to the DMAPI file system while writing to a file that is mirrored in HPSS

HPSS_API_PIO_STRIPE_WIDTH ("1")

Used to control the default simplified PIO stripe width (must be ⇐ HPSS_MAX_STRIPE_WIDTH)

HPSS_API_PIO_BLOCK_SIZE ("1048576")

Used to control the default simplified PIO block size

HPSS_API_PIO_HANDLE_GAPS ("1")

Used to control the default simplified PIO gap handling behavior

HPSS_API_PIO_USE_API_HOST ("0")

Used to control whether simplified PIO contexts will use the HPSS_API_HOSTNAME interface

HPSS_API_REUSE_CONNECTIONS ("0")

Used to control whether TCP/IP connections are to left open as long as an HPSS file is open or are closed after each read or write operation.

HPSS_API_PORT_RANGE (NULL)

Range of TCP/IP ports to use for client api applications (unrelated to HPSS_API_USE_PORT_RANGE). Ex. HPSS_API_PORT_RANGE=X-Y where X and Y are positive integers >= 1024 and ⇐ 65535.

HPSS_API_USE_PORT_RANGE ("0")

Used to control whether HPSS Movers should the configured port range when making TCP/IP connection for read or write operations for the client

HPSS_API_RETRY_STAGE_INP ("1")

Used to control whether retries are attempted on when trying to open files in a Class of Service that is configured for background staging on open

HPSS_API_DISABLE_CROSS_REALM ("0")

Used to control cross-realm traversal

HPSS_API_DISABLE_JUNCTIONS ("0")

Used to control junction traversal

HPSS_API_AUTHN_MECH ("${HPSS_CLIENT_AUTHN_MECH}")

Used to control the select of an authentication mechanism

HPSS_API_RPC_PROT_LEVEL ("${HPSS_RPC_PROT_LEVEL}")

Used to control the select of an RPC protection level

HPSS_API_SAN3P ("on")

Used to control whether SAN3P is on or off.

HPSS_API_TRANSFER_TYPE ("MVRSELECT")

Used to control the transfer type. If set to MVRSELECT then the Mover selects the protocol. If set to TCP, the protocol is forced to TCP/IP.

HPSS_API_OBJ_BUF_LIMIT ("4096")

Maximum number of objects that can be returned.

HPSS_API_XML_BUF_LIMIT ("131072")

Maximum number of xml strings that can be returned.

HPSS_API_XMLOBJ_BUF_LIMIT ("2048")

Maximum number of xml/object entries that can be returned.

HPSS_API_XMLSIZE_LIMIT ("131072")

Maximum length of an XML string buffer.

HPSS_API_XMLREQUEST_LIMIT ("204800")

Maximum length of an XML string request. (xmlsize * number of xml buffers)

HPSS_API_BLOCK_SIZE ("1048576")

Sets the "preferred" block size for efficient filesystem I/O.

HPSS_API_CIPHER ("AES-256-CBC")

Sets the encryption cipher type. Ciphers match OpenSSL naming conventions.

HPSS_API_ENCRYPT ("0")

Used to force use of data encryption

HPSS_API_KEY_SIZE ("32")

The key size in bytes, which is used for PIO API for transfers

31.16. Core Specific

Environment variables related to Core

HPSS_CORE_LARGE_SEG_THRESHOLD ("300")

Defines a "large" tape segment

HPSS_CORE_LOG_XFER_PERF ("0")

Enable (1) or disable (0) internal transfer performance logging

HPSS_CORE_REPACK_OUTPUT ("on")

Control segregation of repack output tapes from other writable tapes

HPSS_CORE_DISKSEGCACHE_MEM_PERCENT ("1.0")

Max percentage of total system memory that can be used by the disk segment cache

HPSS_CORE_TAPE_CACHE_IDLE_TIME ("300")

Tape VV metadata cache retention time in seconds

HPSS_CORE_SEG_CACHE_IDLE_TIME ("60")

Segment metadata cache retention time in seconds

HPSS_CORE_RUMBLE_THREADS ("50")

Number of threads to use for handling rumbler requests

HPSS_CORE_GK_USE_LEGACY_ATTRS ("0")

When a parent directory has the SET_GID bit set, pass the GID for the user instead of GID for the parent directory to the gatekeeper.

HPSS_ACCT_DELAY ("10")

Delay between accounting thread loops.

HPSS_ACCT_RECS_BW_RAW ("100000")

Number of acctlogbandwidth records to select per transaction.

HPSS_ACCT_RECS_CAP_RAW ("100000")

Number of acctlogcapacity records to select per transaction.

HPSS_CORE_CREATE_ALLOW_SUBTYPE ("FALSE")

Skip tape subtype check when creating tape resources. For example, this would allow a LTO9 tape to be created in a storage class defined as LTO8.

HPSS_CORE_SLEEPTIME_BEFORE_ABORT ("1")

Number of seconds to sleep before aborting core server

HPSS_CORE_TAOS_SCHEDULE_PERCENT ("50.0")

Percent of queue that must be unscheduled before calling TAOS.

HPSS_CORE_VERIFY_SECTION_LIMIT ("50")

Number of sections to group together for verify requests

HPSS_RESTRICT_ACCESS_FILE (NULL)

Fully qualified path to the HPSS restrict access file

HPSS_RESTRICT_ACCESS_ERROR_CODE_VALUE ("-13")

Error code to return when user is restricted from an operation.

HPSS_MAX_OPEN_FILES ("20000000")

Hard maximum number of files beyond configured maximum. This is a minimal value for the limit which can be overridden by having a large number of configured open files.

HPSS_MAX_NUM_OPEN_FILES_FOR_AGGR ("4000000")

The maximum number of additional open files allowed for migration.

HPSS_MAX_FILES_IN_AGGR ("50000")

The maximum number of files to be placed in the same aggregate.

HPSS_OFFLINE_OBJECT_STORE_REMINDER_INTERVAL_HOURS ("24")

Controls how often, in hours, the Core server emits a reminder message about Offline object store VVs.

HPSS_OFFLINE_DISK_REMINDER_INTERVAL_HOURS ("24")

Controls how often, in hours, the Core server emits a reminder message about Offline disk VVs.

HPSS_SS_SCHEDULE_BY_CONTROLLER (NULL)

When set, the Core Server will schedule disk I/O requests based on the device "Controller ID" rather than by Mover.

HPSS_CHECKPOINT_CLEANUP_INTERVAL ("120")

Controls how often, in seconds, the background thread that cleans up checkpoints runs. 0 disables background thread

31.17. Logging Specific

Environment variables related to logging

HPSSLOG_PERROR (NULL)

Also log messages to stderr

HPSS_INFRA_LOG_TYPES ("CS_ALARM:CS_EVENT")

Default types of infrastructure messages logged

HPSS_INFRA_LOG_CONF ("${HPSS_PATH_TMP}/$$.log.conf")

The infrastructure logging configuration file

31.18. LS Specific

Environment variables related to LS

LS_DISABLE_LOCAL_CACHE (NULL)

Set to any value to disable the ls client cache file

LS_LOCAL_CACHE_FILENAME (".hpss_ls_cache")

The name of the cache file in the user's home directory

31.19. GHI Specific

Environment variables related to GHI

HPSS_GHI_PATH ("${HPSS_PATH_VAR}/ghi")

Pathname for GHI config files

HPSS_GHI_LOG_PATH ("${HPSS_GHI_PATH}/log")

Pathname for GHI log files

HPSS_GHI_ETC_PATH ("${HPSS_GHI_PATH}/etc")

Pathname for GHI config files

HPSS_GHI_TMP_PATH ("${HPSS_GHI_PATH}/tmp")

Pathname for GHI tmp files

HPSS_GHI_CONF ("${HPSS_GHI_ETC_PATH}/ghi.conf")

GHI config file location

HPSS_GHI_FSCONF ("${HPSS_GHI_ETC_PATH}/ghi_%s.conf")

GHI file system config file

31.20. FTPD Specific

Environment variables related to the FTPD Specific

HPSS_PATH_FTP ("${HPSS_PATH_VAR}/ftp")

FTP daemon etc path

HPSS_FTPD_CONTROL_PORT ("4021")

FTP daemon default control port

HPSS_FTP_RESERVED ("1025")

FTP daemon reserved port IDs

HPSS_FTP_BLOCK_SIZE ("4")

FTP block size (KB)

HPSS_FTP_MAXUSERS ("MAXUSERS")

FTP maximum number of connections

31.21. RAIT Engine Specific

Environment variables related to the RAIT Engine

HPSS_RAIT_BUFFER_COUNT ("6")

Number of buffers used per transfer

HPSS_RAIT_TASK_THREAD_COUNT ("6")

Number of parity threads per transfer

31.22. Migration Purge Server Specific

Environment variables related to the MPS

HPSS_MPS_PURGE_PARALLELISM ("2")

Number of purge worker threads per storage class

HPSS_MPS_MIGR_PURGE_PAUSE_SECS ("300")

Number of seconds to wait before starting migration/purge on MPS startup.

HPSS_MPS_FORCE_MIGRATE_STAGE_BATCH_SIZE ("512")

Batch size to use when force migrate stages files.

31.23. Mover Specific

Environment variables related to the mover

HPSS_MVR_DISABLE_RESERVATIONS (`NULL`)

Disable SCSI reservations for the devices that correspond to the specified comma separated list of HPSS device identifiers.

HPSS_MVR_DEV_PROGRESS_SECS (`NULL`)

Number of seconds between checks for registered updates for device bytes read/written.

HPSS_MVR_LOCATE_BLK_THRESH (`"0"`)

Within this block threshold the tape mover will read the section header rather than locate directly to data

HPSS_MVR_AIO_OPTIMAL_IO_SIZE (`NULL`)

Number of bytes for each AIO request. The default for this is taken from the system block device configuration. If this configuration is not available, then 1MiB is used as the default.

HPSS_MVR_AIO_QUEUE_DEPTH (`NULL`)

Maximum number of pending AIO requests. The default for this is taken from the system block device configuration.

HPSS_MVR_AIO_OP_TIMEOUT (`"60"`)

Number of seconds to wait for an AIO request completion.

HPSS_MVR_DMF_BLOCK_CRC_VALIDATE (`"on"`)

Used to control whether internal block CRC validation for DMF tapes is on or off.

31.24. Security Registry & Service Location Specific

Environment variables related to security

HPSS_SEC_REALM_ADMIN (`"admin/admin"`)

The name of the realm master admin

HPSS_KRB5_AUTHN_MECH (`"krb5"`)

The Kerberos authentication mechanism name

HPSS_KRB5_KEYTAB_FILE (`"auth_keytab:${HPSS_PATH_ETC}/hpss.keytab"`)

Kerberos keytab file for server entities

HPSS_UNIX_AUTHN_MECH (`"unix"`)

The UNIX authentication mechanism name

HPSS_UNIX_AUTHN_SOCKET (`"${HPSS_PATH_VAR}/run/unix.socket"`)

The path to the UNIX socket for the HPSS login service

HPSS_UNIX_AUTHN_ADDRESS ("${HPSS_HOST_FULL_NAME}")

The IP address used by clients to find the HPSS login service

HPSS_UNIX_AUTHN_PORT ("1024")

The TCP port used by the HPSS login service

HPSS_UNIX_KEYTAB_FILE ("auth_keytab:${HPSS_PATH_ETC}/hpss.unix.keytab")

HPSS Unix keytab file name

UNIX_CLNT_KTNAME ("auth_keytab:${HOME}/keytab")

HPSS Unix keytab file name for clients (init)

HPSS_PRIMARY_AUTHN_MECH ("${HPSS_UNIX_AUTHN_MECH}")

The chosen primary authentication mechanism

HPSS_PRIMARY_AUTHENTICATOR ("${HPSS_UNIX_KEYTAB_FILE}")

The authenticator to use for the mechanism

HPSS_CLIENT_AUTHN_MECH ("${HPSS_PRIMARY_AUTHN_MECH}")

The client authentication method to use

HPSS_AUTHZ_SERVICE_CONF ("${HPSS_PATH_ETC}/authz.conf")

File for valid authorization mechanisms

HPSS_SEC_EP_CONF ("${HPSS_PATH_ETC}/ep.conf")

File containing the local endpoints

HPSS_SEC_SITE_CONF ("${HPSS_PATH_ETC}/site.conf")

File containing connect information for local security registry and location service

HPSS_SEC_MUTUAL_AUTH ("TRUE")

Whether to support mutual authentication (true/false)

KRB5_CONFIG ("${HPSS_PATH_SLASH_ETC}/krb5.conf")

Path to the kerberos config file

HPSS_AUTHN_TYPES ("krb5")

Supported authentication types

HPSS_AUTHZ_TYPES ("ldap")

Supported authorization types

HPSS_SITE_LOCATION ("USA")

Location of the HPSS site for internationalization

KRB5_INSTALL_PATH (KRB5_INSTALL_PATH)

Kerberos install path

KRB5_KDC_DIR ("${HPSS_PATH_VAR}/krb5kdc")

Kerberos directory containing local config files for KDC

KRB5_KDC_HOST ("")

Host for Kerberos KDC (just used by mkhpss)

LDAP_INSTALL_PATH (LDAP_INSTALL_PATH)

LDAP library install prefix

HPSS_UNIX_CRED_HOME_DIR ("/home")

HPSS unix credential homedir

HPSS_USE_XREALM_LDAP ("0")

Cross-realm LDAP is allowed

31.25. RPC Specific

Environment variables related to RPC

HPSS_RPC_PORT_RANGE (NULL)

Range of TCP/IP ports to use for RPCs

HPSS_LCG_SERVER_RPC_PORT_RANGE (NULL)

Specific port range to be used by Core and Gatekeeper for clients outside firewall

HPSS_RPC_SOCK_SNDBUF_SZ (NULL)

The RPC socket send buffer size

HPSS_RPC_SOCK_RCVBUF_SZ (NULL)

The RPC socket receive buffer size

HPSS_RPC_SOCK_IO_SZ (NULL)

The RPC socket I/O size to be used

HPSS_RPC_SOCK_NDELAY (NULL)

The RPC socket Nagle setting to be used

HPSS_RPC_PROG_NUM_RANGE ("0x20000000-0x20000200")

The range for RPC program numbers

HPSS_RPC_PROT_LEVEL ("connect")

Default RPC protection level to be used

31.26. SCSI PVR

Environment variables related to SCSI PVR

HPSS_SCSI_DIAG ("off")

Log low level trace to aux log in HPSS_PATH_TMP/pvr.<pid>.diag and turn on SCSI layer diagnostics

HPSS_SCSI_AVOID_CROSS_ZONE (NULL)

Number of times to pause a mount waiting on a pending dismount before going ahead with a cross zone move.

HPSS_SCSI_PVR_INITMAP_THRESH_SECS (NULL)

Number of seconds before the SCSI PVR will attempt to refresh the inventory when an inconsistency is identified

HPSS_SCSI_CMD_QUEUE ("off")

The SCSI Command Queue allows multiple commands to be queued for a SCSI device

31.27. RTM Settings

Settings that modify RTM processing

HPSS_RTM_USE_FULL_PATH ("default")

Control full path reporting in RTM (rtmu) and SSM (HPSS GUI and hpssadm.pl). Valid values include: default - Report full path from root of roots in RTM; Report relative path in SSM. off - Report relative path for both RTM and SSM. on - Report full path from root of roots for both RTM and SSM. Notes: (1) Full path reporting can be a more expensive operation, with an additional request to determine the full path, but caching is used to mitigate follow-on requests. (2) For a change to take effect in SSM (HPSS GUI and hpssadm.pl), the SSM server must be restarted.

31.28. Installation & Misc

Environment variables related to installation

HPSS_PATH_ADM ("${HPSS_PATH_VAR}/adm")

Path where administrative files are placed

HPSS_PATH_CORE ("${HPSS_PATH_ADM}/core")

Path where subsystem core files are placed

HPSS_PATH_TMP ("${HPSS_PATH_VAR}/tmp")

Path where temporary files are placed

HPSS_PATH_ETC ("${HPSS_PATH_VAR}/etc")

Path where runtime config files are placed

HPSS_PATH_RUN ("${HPSS_PATH_VAR}/run")

Path for data specific to running services

HPSS_ENV_CONF ("${HPSS_PATH_ETC}/env.conf")

The path to the environment override file

HPSS_HPSS_CONF ("${HPSS_PATH_ETC}/HPSS.conf")

The path to the HPSS configuration file

HPSS_COPYRIGHT ("${HPSS_ROOT}/copyright")

File containing HPSS copy right info

HPSS_MAGIC_CONF_PATH ("${HPSS_PATH_ETC}/magic.conf")

Path to magic's configuration file

HPSS_PTHREAD_STACK (NULL)

Minimum stack size for HPSS pthreads

HPSS_PTHREAD_STACK_MAX (NULL)

Maximum stack size for HPSS pthreads

HPSS_SAN3P_LINUX_DMMPATH ("off")

Use Linux DM Multipath for SAN3P

31.29. HPSS Group Names

HPSS groups

HPSS_GROUP ("${HPSS_GRP_NAME}")

HPSS group name

HPSS_GROUP_DB ("hpssdb")

HPSS database group name

HPSS_GROUP_LDAP ("ldap")

HPSS group name

HPSS_GROUP_SERVER ("${HPSS_GRP_NAME_SERVER}")

HPSS server group name

HPSS_GROUP_CLIENT ("${HPSS_GRP_NAME_CLIENT}")

HPSS client group name

HPSS_GROUP_SYSTEM ("system")

HPSS group name for local system

31.30. HPSS Group Default IDs

Used by mkhpss to create HPSS associated GROUPS. These may need to be adjusted for NIS environments before running mkhpss/HPSS. All GIDs must be available and consistent across all

server/mover nodes

HPSS_GID (NULL)

HPSS group gid

HPSS_GID_DB (NULL)

HPSS db group gid

HPSS_GID_LDAP (NULL)

HPSS db group gid

HPSS_GID_SERVER (NULL)

HPSS server group gid

HPSS_GID_CLIENT (NULL)

HPSS client group gid

HPSS_GID_SYSTEM ("0")

HPSS group gid for local system

31.31. HPSS User Names

Similar to PRINCIPAL list above, but used by mkhpss to setup additional non-server ids. PRINCIPAL implies DCE/Kerberos, while USER relates more to UNIX based systems.

HPSS_USER_DB ("${HPSS_DB_INSTANCE_OWNER}")

User name for HPSS DB

HPSS_USER_LDAP ("ldap")

User name for LDAP server

HPSS_USER_ROOT ("root")

User name for HPSS local root

31.32. HPSS Server Unix UIDs

Used by mkhpss when setting up HPSS accounts/authentication info. Note that all UIDs must be available and consistent across all server, mover, and VFS client nodes. These must be adjusted before the code is compiled and mkhpss/HPSS is run.

HPSS_UID (NULL)

Unix UID for HPSS

HPSS_UID_DB (NULL)

UNIX UID for HPSS DB2 Instance

HPSS_UID_LDAP (NULL)

UNIX UID for HPSS LDAP

HPSS_UID_ROOT ("0")

UNIX UID for HPSS local root

31.33. HPSS Object Store Encryption Master Key File

Currently used for object store identities secret key

HPSS_OBJSTOR_MASTER_EK_FILE ("objstor.master.key")

Name of file containing object store master encryption key

31.34. HPSS Object Store Notifcation Configuration

Configuration for object store restore notifications.

HPSS_OBJSTOR_SNS_TOPIC_NAME ("hpss-restore-notify")

AWS SNS topic for restore notifications

HPSS_OBJSTOR_SNS_CA_LOCATION (NULL)

Notification service certificate location

Appendix E: The /var/hpss files

The `/var/hpss` directory tree is the default location of a number of HPSS configuration files, log files, and other files needed by the servers.

The directories and configuration files can be created with the **mkhpss** utility or hand-created. Be very careful when using **mkhpss** utility as selecting the wrong option can damage the already partially configured HPSS system. The log files and other files are created by the servers.

The directories in `/var/hpss` include:

acct. The usual directory to hold accounting report files and checkpoint files. This location is specified in the accounting policy. The report and checkpoint files are created by the accounting report utility; see the Generating an accounting report section. The administrator must regularly remove unneeded reports to prevent the `/var/hpss` file system from filling.

adm. This directory contains one log file and one subdirectory:

- **hpssd.failed_server.** A file of log entries for server startups and terminations. Created and maintained by the Startup Daemon.
- **core**. The default directory where HPSS servers put "core" files if they terminate abnormally. The system administrator must clean out old core files regularly to avoid filling up `/var/hpss`.

cred. The default directory where Kerberos credential files for HPSS servers are placed.

doc. The default directory where the HPSS documentation is installed.

etc. The default directory where many UNIX configuration files are placed. These files include:

- **HPSS.conf.** A configuration file for ftp and other HPSS client applications. Can be initialized from the template file `/opt/hpss/config/templates/HPSS.conf.tmpl`. See HPSS.conf(7) and `HPSS.conf` configuration file for more details.
- **authz.conf.** Created by **mkhpss**. Lists HPSS SEC libraries and functions for performing initialization of authorization manager: LDAP or UNIX. Can be initialized from the template file `/opt/hpss/config/templates/authz.conf.template`. See the Security services configuration section for details.
- **env.conf.** Created as an empty file by **mkhpss**. Contains site-specific environment variables for HPSS. Some utilities and servers in HPSS may automatically source this file upon startup for necessary environment variables. For others, environment variables like HPSS_PATH_VAR may need to be defined first.
- **ep.conf.** Contains the endpoints of the HPSS location service. Used by HPSS client application programs to determine the address at which to contact HPSS. Created by the utility `/opt/hpss/config/hpss_bld_ep`. This utility should be executed by root after the Core Server has been configured. It takes no arguments.
- **hpss.keytab.** Created by **mkhpss**. Contains the username and password for HPSS servers to use for Kerberos authentication with a keytab file. Can be created manually by using **ktutil** to add

each of the HPSS server principals to the file. See your Kerberos documentation for the usage of **ktutil**.

- **hpss.unix.keytab.** Created by **mkhpss**. Contains the username and password for HPSS servers to use for UNIX authentication with a keytab file. Can be created manually by using the hpss_unix_keytab(7) utility to add each of the HPSS server principals to the file. See the HPSS man page for the usage of hpss_unix_keytab(7).
- **mm.keytab.** Created by **mkhpss**. The keytab for user *hpss*. Used by utilities which read the DB2 databases. Can be created manually with the `/opt/hpss/config/hpss_mm_keytab` utility. The syntax is

  ```
  hpss_mm_keytab -f /var/hpss/etc/mm.keytab hpss
  ```

  The program will prompt for the UNIX password of user *hpss*. Can be created manually with the `/opt/hpss/config/hpss_mm_keytab` utility. The syntax is

  ```
  hpss_mm_keytab -f /var/hpss/etc/mm.keytab hpss
  ```

  The program will prompt for the UNIX password of user *hpss*.

- site.conf(7). Created by **mkhpss**. Contains the site name, realm name, realm ID, authorization mechanism and authorization URL to utilize. Additionally contains encryption specifications. Can be initialized from the template file `/opt/hpss/config/templates/site.conf.template`. See the Security services configuration section and the site.conf man page for details.
- **rc.db2**. Created by **mkhpss**. Script for starting DB2. Can be initialized from the template file `/opt/hpss/config/templates/rc.db2.template`. Change the line in the file:

  ```
  export DB2INSTANCE="%<DB2INSTANCE>%"
  ```

  to:

  ```
  export DB2INSTANCE="hpssdb"
  ```

  or to whatever your site has named your DB2 instance.

- **rc.krb**. Created by **mkhpss**. Script for starting the Kerberos servers. Can be initialized from the template file `/opt/hpss/config/templates/rc.krb5.template`. (Looks like you have to replace the lines for KRB5_INSTALL_PATH and KRB5_KDC_PROFILE in here.)
- **passwd.** Created by **mkhpss**. A local HPSS-only password file for use with UNIX authentication and authorization. Optionally, the system password file can be used instead. If a lot of users are added or removed from the system, it may be necessary to remove backups of this file generated by the HPSS user management utilities.
- **group.** Created by **mkhpss**. A local HPSS-only group file for use with UNIX authentication and authorization. Optionally, the system group file can be used instead. If a lot of users are added

or removed from the system, it may be necessary to remove backups of this file generated by the HPSS user management utilities.

- **shadow.** Created by **mkhpss**. A local HPSS-only shadow file for use with UNIX authentication and authorization. Optionally, the system shadow file can be used instead. If a lot of users are added or removed from the system, it may be necessary to remove backups of this file generated by the HPSS user management utilities.
- **kt.shadow.** Created by **mkhpss**. A local HPSS-only secret stash for UNIX keytabs. Both system UNIX and HPSS UNIX use this file.
- **hpss_san3p.conf.** Contains fiber connectivity information for SAN3p devices and clients. This file typically resides on a Mover machine and is used to indicate the association between client machines, using IP addresses, and SAN3p devices. A template can be found in the `./config/templates/hpss_san3p.conf.template` file. See the instructions in the template for details about how to update the file for specific SAN configurations. See the Device and drive management section for more information on SAN3p devices.

ftp. Files for FTP configuration. See the FTP/PFTP daemon configuration section for more information.

gk. The recommended directory for the site policy configuration file used by the Gatekeeper if the Gatekeeper is configured to do gatekeeping services. The file is usually named `gksitepolicy`. It is created by the system administrator. See Gatekeeper.

hpssdb. The directory to hold the DB2 instance configuration information and *CFG* database tables.

krb5kdc. The directory for Kerberos configuration files. These include:

- **kadm5.keytab**. The keytab for the Kerberos administrative user
- **kdc.conf**. Kerberos configuration file. See the Kerberos documentation. See also the template file `/opt/hpss/config/templates/kdc.conf.template`.

log. By default, HPSS log files should reside in `/var/hpss/log`. Because HPSS logs to syslog, the system syslog configuration must be updated to place HPSS logs in this location. Run the `configure_logging` tool during installation (see Configure Logging) to automatically configure rsyslog. See the Logging overview section for details.

mps. The directory in which the MPS places its migration/purge reports, if it is configured to produce these reports. Reports are generated every 24 hours. The system administrator will need to remove these reports regularly when no longer needed to avoid filling up the `/var/hpss` file system.

ssm. The usual directory for SSM configuration files. These include:

- A file to store alarms and events if SSM is configured to buffer alarm and event messages in a disk file (as opposed to keeping them in memory). This pathname is defined by the HPSS_SSM_ALARMS environment variable.
- An optional subdirectory to hold keytabs for hpssadm(7) users on the system. The directory and keytab files are created by the system administrator.
- **login.conf**. (Optional) Contains information required by SSM authentication. A copy of this file

is included in /opt/hpss/bin/hpss.jar and it should need no customization. However, a template is provided in /opt/hpss/config/templates/login.conf.template should sites need it.

- **ssm.conf.** Configuration file for SSM client programs (hpssgui(7) and hpssadm(7)). Can be initialized from the template file /opt/hpss/config/templates/ssm.conf.template. See the instructions inside the template file.

See the Using SSM section for details.

tmp. The Startup Daemon uses /var/hpss/tmp as the default location for HPSS server lock files, creating a lock file for each server it starts on the node.

HPSS may also write diagnostic log files and disk allocation maps in this directory, when configured to do so. The lock files are very small, but the log files and maps can be very large.

The diagnostic files are named gasapi.diag, secapi.diag, or <pid>.diag. These files can be truncated to free up the disk space they occupy, or they can be removed from the system.

As HPSS runs, it checks whether /var/hpss/tmp/gasapi.diag and /var/hpss/tmp/secapi.diag exist. If one of them does, HPSS processes will begin appending diagnostic data to the file and continue to do so as long as they continue running.

To truncate one of the .diag files (but leave the process writing it running, and possibly writing more data to the file in the future), simply redirect the output of a silent command (that is, one that produces no output) to the file:

```
% /bin/echo -n "" > /var/hpss/tmp/secapi.diag
```

To remove a .diag file completely so that HPSS processes will no longer write diagnostic data to it, it is necessary to shut down all processes that have it open for writing. For secapi.diag and gasapi.diag, this will typically mean shutting down all of HPSS, then removing the file. Once all the standard HPSS servers are shut down through SSM:

```
% rc.hpss -m stop
% rc.hpss -d stop
% rm /var/hpss/tmp/secapi.diag /var/hpss/tmp/gasapi.diag
```

Per-process diagnostic files are generated by sending a HUP signal to the process of interest:

```
% kill -HUP <pid>
```

To truncate <pid>.diag and free up the disk space it occupies:

```
% /bin/echo -n "" > /var/hpss/tmp/<pid>.diag
```

To remove <pid>.diag completely, determine which server is running with that process id and shut

it down, then:

```
% rm /var/hpss/tmp/<pid>.diag
```



If a `.diag` file is simply removed using **rm** while a process still has it open, the directory entry disappears but the disk space occupied is not released. Thus, it is possible to wind up with a disk full of unreachable and unreleasable data. The only way to recover is to find the process that has the phantom files open and shut them down. Once this is done, the orphaned space will be released and returned to the system for reuse.

Appendix F: Man Pages

The following appendix contains man pages for HPSS tools. These can be viewed on systems where HPSS has been installed. See Manual pages setup for local system installation details.

F.1. crtfileset(7)

F.1.1. Name

crtfileset - create an HPSS-Only fileset

F.1.2. Syntax

crtfileset -s *subsystem_id*
[-i *fileset_id*] [-l] [-f *family*] [-c *cos*]
[-p *principal*] [-k] [-t *keytab*] [-a *mech*]
[-n *name*]

F.1.3. Parameters

HPSS-Only REQUIRED ARGUMENTS:

-s

Subsystem for fileset creation (ex: 1)

HPSS-Only OPTIONAL ARGUMENTS:

-i

Fileset ID (ex: 1076341225.1079029474)

-f

Family ID (ex: 0)

-c

Class of Service (ex: 1) [default 0]

-p

Principal (ex: hpssssm)

-l

New login credential

-k

Use keytab

-t

Authenticator (ex: /var/hpss/etc/hpss.keytab)

-a

Authentication mechanism (ex: krb5)

-n

Fileset name (ex: HPSSOnlyFileset)

F.1.4. Description

This command creates an HPSS-Only fileset. It was previously known as create_fset.

Creating an HPSS-Only fileset:

The Fileset ID can be specified either in the form <hi>.<low> or <hi>,,<low>, where <hi> and <low> are integers.

The Subsystem ID is an integer representing in which subystem the fileset should be created. If only one subsystem exists, the Subsystem ID is likely 1.

The Fileset Name must be a string of 127 characters or less. If a fileset name is not supplied, a random name will be generated, which begins with crtfileset_Fileset.

F.1.5. Examples

This example will create an HPSS-Only fileset on subsystem 1. The fileset will be named FunWithFilesets.

```
crtfileset -s 1 -i 4076241205.1089029475 -n FunWithFilesets
```

F.1.6. See Also

crtjunction, lsfilesets

F.2. crtjunction(7)

F.2.1. Name

crtjunction - Create a junction to a file set.

F.2.2. Syntax

crtjunction [-a *auth-type*] [-l] [-p *user*] [-k]
[-t *keytab*] [-f *fileset name*] [-j *junction path*]

F.2.3. Description

crtjunction can be used to create junctions to filesets. crtjunction only creates junctions to HPSS filesets.

crtjunction can be used to create junctions to any type of fileset.

Only the root user or a *trusted* principal is allowed to create junctions. A trusted principal is any principal having Write permission in an ACL entry listed on the Core Server's security object ACL.

F.2.4. Parameters

-a
Used to specify the authentication type to use Supported values are: krb5 (Kerberos) unix (Unix)

-l
Used to create a new login credential

-p
Used to specify a principal name

-k
Used to specify login using a keytab file

-t *keytab*
Used to specify the keytab file path

-f *fileset name*
Used to specify the fileset name

-j *junction path*
Used to specify the full junction path

F.2.5. Example

The following command will create a junction at "/:/fileset.0" that points to the fileset "fileset.0".

crtjunction -f fileset.0 -j /:/fileset.0

The following command will create a junction at "/:/fileset.0" that points to the fileset "fileset.0" using unix authentication and the default keytab with a trusted user name.

crtjunction -a unix -k root -p hpssssm -f fileset.0 -j /:/fileset.0

F.2.6. See Also

deljunction

F.3. deljunction(7)

F.3.1. Name

deljunction - Delete a junction.

F.3.2. Syntax

deljunction [-a *auth-type*] [-l] [-p *user*]
[-k] [-t *keytab*] [-j *junction path*]

F.3.3. Description

deljunction is used to delete junctions. deljunction can only delete the junctions in the HPSS name space.

Only the root user or a *trusted* principal is allowed to delete junctions. A trusted principal is any principal having Write permission in an ACL entry listed on the Core Server's security object ACL.

F.3.4. Parameters

-a

Used to specify the authentication type to use Supported values are: krb5 (Kerberos) unix (Unix)

-l

Used to create a new login credential

-p

Used to specify a principal name

-k

Used to specify login using a keytab file

-t *keytab*

Used to specify the keytab file path

-j *junction path*

Used to specify the full junction path

F.3.5. Example

The following command will delete a junction at (/:/fileset.0).

deljunction -j /:/fileset.0

The following command will delete a junction at (/:/fileset.0) using unix authentication and the default keytab with a trusted user name.

deljunction -a unix -k root -p hpssssm -j /:/fileset.0

F.3.6. See Also

crtjunction

F.4. device_scan(7)

F.4.1. Name

device_scan - Display list of scsi devices.

F.4.2. Syntax

device_scan [-VvDdij?h] [-t *disk*|*tape*|*changer*] [-s *dev*]

F.4.3. Description

device_scan is a utility that lists scsi devices and their associated inquiry or read element status information. It can list information for medium changer, tape, and disk devices.

device_scan should be run when configuring a SCSI PVR on a system to determine the Command Device field. This will be listed as the "Device" (e.g. /dev/sg14) . Additionally, serial numbers and element addresses may be obtained for the library's drives and can be useful in configuring the SCSI PVR devices.

The default output mimics the same information as the original device_scan but with different formatting. Additional inquiry information can be obtained via the use of different command line options.

This tool must be run as root.

F.4.4. Options

-t

Type of SCSI device to scan for. The default is "changer". The other types are "disk" and "tape".

-i

Interrogate drives and obtain their element addresses and serial numbers. This option is only compatible with the "changer" type.

-j

Output in JSON format.

-s

Scan a single device.

-?

Print usage information.

-h

Print usage information.

-v

Verbose - this will print all of the standard inquiry information.

-V

Very verbose - this will print all of the standard inquiry information plus a hex dump of the additional inquiry length.

-d

Print out debug output. This will display intermittent error messages and SCSI status/sense information.

-D

Print out a lot of debug output. This will display a hex dump of SCSI command structures and return data.

F.4.5. Examples

```
$ device_scan

1) Device=[/dev/sg22]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585 0]
        Vital Product   =
               Identifier  =   ['T10:IBM|03584L22        0000000115850405']
2) Device=[/dev/sg20]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585 0]
        Vital Product   =
               Identifier  =   ['T10:IBM|03584L22        0000000115850405']
3) Device=[/dev/sg14]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585]
        Vital Product   =
               Identifier  =   ['T10:IBM|03584L22        00000001158504CB']
4) Device=[/dev/sg16]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585]
        Vital Product   =
               Identifier  =   ['T10:IBM|03584L22        00000001158504CB']
```

In this example you can tell that there are four devices. You can also tell that there are two separate libraries because the there are only two unique "Identifier" fields.

```
$ device_scan -i

1) Device=[/dev/sg22]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585 0]
        Vital Product   =
               Identifier  =   ['T10:IBM|03584L22        0000000115850405']
                Data Transfer Elements
                                Address Serial
                1           =   [257]   [not supported]
                2           =   [258]   [not supported]
                3           =   [259]   [not supported]
                4           =   [260]   [not supported]
2) Device=[/dev/sg20]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585 0]
        Vital Product   =
               Identifier  =   ['T10:IBM|03584L22        0000000115850405']
                Data Transfer Elements
                                Address Serial
                1           =   [257]   [not supported]
                2           =   [258]   [not supported]
                3           =   [259]   [not supported]
                4           =   [260]   [not supported]
3) Device=[/dev/sg14]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585]
        Vital Product   =
               Identifier  =   ['T10:IBM|03584L22        00000001158504CB']
                Data Transfer Elements
                                Address Serial
                1           =   [269]   [not supported]
                2           =   [270]   [not supported]
                3           =   [271]   [not supported]
                4           =   [272]   [not supported]
4) Device=[/dev/sg16]
        Vendor      =   [IBM]
        Product     =   [03584L2]
        Revision    =   [736]
        Vendor info =   [736313000000011585]
        Vital Product   =
```

```
                  Identifier  =   ['T10:IBM|03584L22        00000001158504CB']
                   Data Transfer Elements
                                  Address Serial
                   1           =   [269]   [not supported]
                   2           =   [270]   [not supported]
                   3           =   [271]   [not supported]
                   4           =   [272]   [not supported]
```

The -i option allows the inclusion of data transfer element information. As you can see, each of the two libraries has four drives. Neither library supports getting serial number information on those drives.

```
$ device_scan -j
{
 "Devices": [
  {
   "Path": "/dev/sg5",
   "Vendor": "STK",
   "Product": "L700",
   "Revision": "0104",
   "VendorInfo": "010",
   "PeripheralQualTypeString": "CONNECTED",
   "PeripheralQualTypeRaw": 0,
   "PeripheralDeviceTypeString": "CONNECTED",
   "PeripheralDeviceTypeRaw": 8,
   "DeviceTypeQualifier": 0,
   "RemovableMedia": "supported",
   "ISOVersion": 0,
   "ECMAVersion": 0,
   "ANSIVersion": 5,
   "ResponseFormat": 2,
   "AdditionalLength": 59,
   "16BitWideBus": "supported",
   "Sync": "not supported",
   "CommandQueue": "not supported",
   "VitalProductID": "T10:STK|L700            TESTTEST  "
  }
 ]
}
```

F.4.6. See Also

scsi_home

F.5. dump_acct_sum(7)

F.5.1. Name

dump_acct_sum - report number of files per class of service

F.5.2. Syntax

dump_acct_sum
[-g *Global_DB_Name*]
[-h *Database_Schema_Name*]
[-s *Subsystem_ID*]
[-v]
[-b]
[-c]
[-S]
[-j]
[-a]

F.5.3. Description

dump_acct_sum reports the total number of files in each class of service. It obtains the information by reading the account capacity and bandwidth summary tables.

If no arguments are given, a summary of the account capacity and bandwidth tables is displayed. This is the equivalent of the -S option.

dump_acct_sum can be used to verify that a COS is empty before deleting the COS configuration.

F.5.4. Parameters

-g *Global_DB_Name*

The global configuration database Name. The default value is taken from the environment variable HPSS_GLOBAL_DB_NAME.

-h *Schema_Name*

The schema database Name. The default value is taken from the environment variable HPSS_MM_SCHEMA_NAME.

-s *Subsystem_ID*

Subsystem Id. The default is 1.

-v

Displays additional information about errors.

-j

Output in JSON format.

-c

Display account capacity breakdown table.

-b

Display account bandwidth breakdown table.

-S

Display accounting capacity and bandwidth summary tables. This is the default behavior if no options are selected.

-a

Only report COSes that are currently active (i.e., not empty). However, if a COS is inactive but has recorded bandwidth activity since the last run of the hpss_acct accounting report program, that bandwidth activity will still be displayed.

F.5.5. Examples

```
$ dump_acct_sum -s 1

                  ================================================
                     Summary report for subsystem 1
                  ================================================
   ================================================================
           Capacity Summary By Class Of Service
   ================================================================
   COS 1, count = 19, bytes = 0
   COS 2, count = 589164, bytes = 10782228092
   COS 3, count = 10004997, bytes = 11053572
   COS 4, count = 167, bytes = 10739836854
   COS 5, count = 4, bytes = 4396679168
   COS 6, count = 0, bytes = 0
   COS 7, count = 0, bytes = 0
   total files stored = 10594351
   total bytes stored = 25929797686


   ================================================================
           Bandwidth Summary By Class Of Service
   ================================================================
   COS 2, Bytes Read = 555533206177, Bytes Written = 130012886870
   COS 3, Bytes Read = 1048576, Bytes Written = 12097562
   COS 4, Bytes Read = 81590214171, Bytes Written = 178614816198
   COS 5, Bytes Read = 7560232960, Bytes Written = 6124732416
   COS 6, Bytes Read = 0, Bytes Written = 209715200
   COS 7, Bytes Read = 10741622784, Bytes Written = 131391938560
   total bytes written = 446366186806
   total bytes read = 655426324668
   total files written = 10840903
```

```
total files read = 6449838
```

```
$ dump_acct_sum -c

            ================================================
               Accounting Summary report for subsystem 1
            ================================================
=======================================================
RealmId      = 0
AccountId    = 0
UID          = 206
GID          = 202
COSId        = 101204
LastModTime  = Sun 06 Apr 2155 07:50:32 PM EDT
FilesStored  = 1
BytesStored  = 2097152
=======================================================
RealmId      = 0
AccountId    = 0
UID          = 213
GID          = 202
COSId        = 101204
LastModTime  = Sun 06 Apr 2155 07:50:02 PM EDT
FilesStored  = 1
BytesStored  = 3145728
=======================================================
RealmId      = 0
AccountId    = 0
UID          = 206
GID          = 202
COSId        = 105101
LastModTime  = Mon 25 Feb 2019 06:20:11 PM EST
FilesStored  = 0
BytesStored  = 0
=======================================================
RealmId      = 0
AccountId    = 0
UID          = 213
GID          = 202
COSId        = 105101
LastModTime  = Mon 25 Feb 2019 07:27:26 PM EST
FilesStored  = 0
BytesStored  = 0
=======================================================
RealmId      = 0
AccountId    = 0
UID          = 206
GID          = 202
COSId        = 106101
LastModTime  = Mon 25 Feb 2019 11:14:25 AM EST
```

```
    FilesStored  = 0
    BytesStored  = 0
    ======================================================
    RealmId      = 0
    AccountId    = 0
    UID          = 213
    GID          = 202
    COSId        = 106101
    LastModTime  = Fri 04 Apr 2155 05:07:15 AM EDT
    FilesStored  = 1
    BytesStored  = 2097152
    ======================================================
    RealmId      = 0
    AccountId    = 213
    UID          = 213
    GID          = 202
    COSId        = 105101
    LastModTime  = Mon 25 Feb 2019 11:34:40 AM EST
    FilesStored  = 0
    BytesStored  = 0
```

This shows the same information as example 1, plus additional details, including the account id, uid, and gid of the space used in each COS.

F.6. dumpbf(7)

F.6.1. Name

dumpbf - List the bitfile segments for a file pathname.

F.6.2. Syntax

dumpbf -f *File Pathname*
[-p *Utility Principal Name*]
[-k *Utility Authenticator*]
[-g *Global DB Name*]
[-m *Utility Security Mechanism*]
[-s *HPSS schema name*]

F.6.3. Description

The "dumpbf" HPSS utility will print an ordered set of bitfile segments by Storage Type and byte offset within bitfile for the input file pathname. This routine will connect to the core server in order to retrieve the bitfile Id for the specified pathname. The bitfile segments will be read from metadata and sorted as they are read and placed in a list. After all the segments have been read and sorted, the results are sent to standard output. Included in the output for a segment which resides on a tape virtual volume will be the physical volume name(s).

F.6.4. Options

-f *File Pathname*

The pathname of the file to dump.

-g *Global DB Name*

The name of the global database. If not present, the value for this parameter will be taken from the environment variable HPSS_GLOBAL_DB_NAME.

-p *Principal Name*

The server principal name which this utility will run under. Usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not present, the value for this parameter will be taken from the HPSS_PRINCIPAL_SSM environment variable.

-k *Authenticator*

The authenticator which this utility will use to get authenticated. Usually run as SSM. The Keytab name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not present, the value for this parameter will be taken from the HPSS_PRIMARY_AUTHENTICATOR environment variable.

-s *Schema Name*

The schema name associated with the HPSS Databases. If not present, the value for this parameter will be taken from the environment variable HPSS_MM_SCHEMA_NAME.

-m *Security Mech*

The security authentication mechanism that this utility will use. If not present, the value for this parameter will be taken from the environment variable HPSS_PRIMARY_AUTHN_MECH.

F.6.5. Example

The following command will list the segments of the file "/dsf/Home0508". The file resides on a two wide tape stripe volume.

```
        $ dumpbf -f /dsf/Home0508 -g hpss_global_db
                  -p hpssssm -k /var/hpss/etc/hpss.keytab -m krb -s hpss


         Bitfile segment dump of: /dsf/Home0508
         Bitfile Id: Object UUID 003785b6-26b1-1191-ab55-02608c2facae
         ServerDependent 00743a0658eb1005a821000000000004

                     **Tape Segments**:
         StorageClass: 3
         Number of segments at this level = 1
         Total length of segments at this level = 90573824

         Byte Offset: 0
         Segment Length: 90573824   PV(s):  JIM00200    JIM00400
```

F.6.6. See Also

lsvol

F.7. dumppv_pvl(7)

F.7.1. Name

dumppv_pvl - List the physical volumes managed by a PVL.

F.7.2. Syntax

dumppv_pvl [-p *Utility Principal Name*]
[-k *Utility Authenticator*]
[-g *HPSS global config database name*]
[-m *Authentication Mechanism*]
[-s *Database schema name*]
[-j]
[-r]
[-u]

F.7.3. Description

This utility will produce a list of the physical volumes managed by the PVL. The list will include the name, media type and the Storage Class(if allocated) of the volume.

If certain command line arguments are not specified, HPSS environment variables will be used (if set).

F.7.4. Options

-g *Global Name*

The global configuration database Name. file. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

-s *Schema Name*

The schema database Name. file. If set, the environment variable HPSS_MM_SCHEMA_NAME will be used if an argument is not specified on the command line.

-p *Principal Name*

The server principal name which this utility will run under. The default value is picked up from the environment variable HPSS_PRINCIPAL_SSM.

-k *Authenticator*

The authenticator which this utility will authenticate with. The default value is picked up from the environment variable HPSS_PRIMARY_AUTHENTICATOR.

-m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

-j

Output in JSON format.

-r

Display PVR name (or N/A) of the volume.

-u

Only volumes in unallocated state will be reported.

F.7.5. Example

The following command will list the physical volumes managed by the PVL.

```
$ dumppv_pvl -g cfg -s hpss -p hpss_ssm -k /krb5/hpss.keytabs
```

OR (using environment variables)

```
$ dumppv_pvl

PVL Physical Volumes:

      Name                    Type           Class          State
      CE908500              9940B Tape       405            Busy
      CE908600              9940B Tape       405            Idle
      CE908700              9940B Tape       200            Idle
      DSK01000           Default Disk        1              Busy
      DSK01100           Default Disk        1              Busy
      DSK01200           Default Disk        1              Busy
      DSK01300           Default Disk        1              Busy
      DSK11000           Default Disk        10             Idle
      DSK11100           Default Disk        10             Idle
      DSK11200           Default Disk        10             Idle
      DSK11300           Default Disk        10             Idle
      GENERI00      Unknown media type       unallocated    Idle
      ST000100           Default Disk        101            Busy
      ST000200           Default Disk        101            Busy
      ST000300           Default Disk        102            Busy
      ST000400           Default Disk        102            Busy
      ST000500           Default Disk        104            Busy
      ST000600           Default Disk        104            Busy
      ST000700           Default Disk        105            Busy
      ST000800           Default Disk        105            Busy
      ST000900           Default Disk        105            Busy
      ST001000           Default Disk        105            Busy
      ST001100           Default Disk        103            Idle
      ST001200           Default Disk        103            Idle
      SY013300           3590 SL Tape        300            Idle
      SY013700           3590 SL Tape        404            Idle
```

```
      SY013800            3590 SL Tape        404             Idle
      SY013900            3590 SL Tape        404             Idle
      SY014000            3590 SL Tape        401             Idle
      SY014100            3590 SL Tape        403             Idle
      SY014200            3590 SL Tape        403             Idle
      SY014300            3590 SL Tape        403             Idle
      SY014400            3590 SL Tape        402             Idle
      SY014500            3590 SL Tape        402             Idle
      SY014600            3590 SL Tape        402             Idle
      SY014700            3590 SL Tape        402             Idle
      UN515900            3590 SL Tape        401             Idle
```

F.7.6. See Also

dumppv_pvr, lsvol

F.8. dumppv_pvr(7)

F.8.1. Name

dumppv_pvr - List the physical volumes managed by a PVR.

F.8.2. Syntax

dumppv_pvr -l *PVR Generic Descriptive name*

dumppv_pvr -n
[-g *HPSS Global Config Database Name*]
[-s *HPSS Database Schema Name*]
[-p *Utility Principal Name*]
[-k *Utility Authenticator*]
[-m *Authentication Mechanism*]
[-j]

F.8.3. Description

This utility will produce a list of the physical volumes managed by the PVR. The output includes a count of the number of mounts performed on a cartridge since it was placed in service and the status of the mount. The output may also include a list of physical volumes pending failed move operation for which the move operation needs to be retried (if this condition exists).

Additionally, the output will include whether the PVR is currently disabled.

If certain command line arguments are not specified, HPSS environment variables will be used (if set).

F.8.4. Options

-l *PVR Desc. Name*

The descriptive name of the HPSS PVR Server which manages the volumes to be scanned. The descriptive name of a PVR server can be found from SSM through: Monitor→Servers screen.

-n

This option is used to list the physical volumes pending a move operation for which the volumes currently aren't assigned to a PVR; this condition may occur on failed move operations that need to be retried. There may be no volumes in this state in which case no output will occur. The -l option is not required when using the -n option. The -n option is not required when using the -l option; the -l option will also output the list obtained by using the -n option.

-g *Global Name*

The global configuration database Name. file. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

-s *Schema Name*

The schema database Name. If set, the environment variable HPSS_MM_SCHEMA_NAME will be used if an argument is not specified on the command line.

-p *Principal Name*

The server principal name which this utility will run under. The default value is picked up from the environment variable HPSS_PRINCIPAL_SSM.

-k *Authenticator*

The authenticator which this utility will authenticate with. The default value is picked up from the environment variable HPSS_PRIMARY_AUTHENTICATOR.

-m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

-j

Output in JSON format.

F.8.5. Example

The following command will list the physical volumes managed by the PVR "STK PVR".

```
$ dumppv_pvr -l "STK PVR" -g cfg -s hpss
          -p hpss_ssm -k /krb5/hpss.keytabs

                            OR (using environment variables)

$ dumppv_pvr -l "STK PVR"

PVR 'STK PVR' Physical Volumes:

        Name    Total Mounts    Mount Status
        000060       118         DISMOUNTED
        000061       113         DISMOUNTED
        000062       101         DISMOUNTED
        000063       110         DISMOUNTED
        000064        96         DISMOUNTED
        000065         2         DISMOUNTED
        000066         4         DISMOUNTED
        000067         7         DISMOUNTED
        000068        57         DISMOUNTED
        000069         2         DISMOUNTED
        000070        17         DISMOUNTED
        000071        43         DISMOUNTED
        000072         0         DISMOUNTED
        000073         0         DISMOUNTED
        000074         0         DISMOUNTED
        000075         0         DISMOUNTED
```

```
        000076        0         DISMOUNTED
        000077        0         DISMOUNTED
        000078        0         DISMOUNTED
        000079        0         DISMOUNTED
        GENERI        0         CHECKED OUT

Disabled: False

Physical Volumes Pending Move (not assigned to a PVR); Move Retry Required:

        Name     Total Mounts   Mount Status
        BOG715        7         MOVE PENDING
        BOG716        6         MOVE SHELF PENDING

Disabled: False

$ dumppv_pvr -n

Physical Volumes Pending Move (not assigned to a PVR); Move Retry Required:

        Name     Total Mounts   Mount Status
        BOG715        7         MOVE PENDING
        BOG716        6         MOVE SHELF PENDING

Disabled: False
```

F.8.6. See Also

dumppv_pvl, lsvol

F.9. dump_sspvs(7)

F.9.1. Name

dump_sspvs - List physical volumes managed by a Core Server

F.9.2. Syntax

dump_sspvs
[--subsystem-id=*ss*]
[--help]
[--authentication-mechanism=*am*]
[--authenticator-name=*an*]
[--db-schema-name=*sn*]
[--exclude-non-accessed]
[--global-db-name=*dn*]
[--login-principal=*lp*]
[--lstapes-format]
[--lstapes-format-units=*mb*]
[--select-disk-volumes]
[--select-object-volumes]
[--select-file-family=*fm*]
[--select-locked-volumes]
[--select-migration-output]
[--select-storage-class=*sc*]
[--show-aggregate-stats]
[--show-pvr-id]
[--show-pvr-name]
[--show-priority]
[--show-map-state]
[--show-creation]
[--json]
[--vv-name]

OR

dump_sspvs
[-i *ss*]
[-h]
[-m *am*]
[-k *an*]
[-s *sn*]
[-x]
[-g *dn*]
[-p *lp*]
[-t]
[-T *mb*]
[-d]

[-o]
[-f *fm*]
[-l]
[-M]
[-c *sc*]
[-a]
[-r]
[-R]
[-P]
[-n]
[-C]
[-j]
[-v]

F.9.3. Description

This program scans the physical volumes managed by the specified Core Server and outputs the physical volume name, VV condition, family ID, storage class and usage statistics for the volume. A summary of available and used bytes by storage class is also output. Volumes may be selected by name, family, storage class and by a number of status related options.

Values for the subsystem ID, global database name, schema name, authentication mechanism, authenticator and principal name will be taken from the user's environment if the corresponding arguments are not supplied on the command line. If these environment variables do not appear in the user's environment, default values will be used. The command line arguments override the environment.

Note that this program is locale aware and will output numerical values according to your shell's configured locale. If numerical output is formatted in unexpected ways, ensure that your environment's LC_ALL variable is configured appropriately.

F.9.4. Options

--authentication-mechanism=*am*

-m *am*

The security mechanism used to authenticate the client. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--authenticator-name=*an*

-k *an*

The authenticator which is used to obtain credentials. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--db-schema-name=*sn*

-s *sn*

The ASCII name of the Db2 schema that contains the global database. If not specified on the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--exclude-non-accessed

-x

Exclude tapes that have not been accessed. A tape is considered to have been accessed if it has ever been read or written. This option is invalid when dumping disk or object store volumes.

--global-db-name=*dn*

-g *dn*

The ASCII name of the global database where the basic configuration information for HPSS is found. If not specified on the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--help

-h

Show a brief statement on how to run dump_sspvs.

--login-principal=*lp*

-p *lp*

The server principal name under which this program will run. Dump_sspvs is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided on the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--lstapes-format

-t

Generate lstapes format output. This option is invalid when dumping disk or object store volumes.

--lstapes-format-units=*mb*

-T *mb*

Identical to -t option but with a byte units specification. Options are b, kb, mb, gb, tb, pb, eb.

--select-disk-volumes

-d

Dump disk PVs rather than tape PVs. The -d and -o options cannot be used together.

--select-object-volumes

-o

Dump object store PVs rather than tape PVs. The -o and -d options cannot be used together.

--select-file-family=*fm*

-f *fm*

If the family ID is specified, the output will be restricted to volumes in that family. This option is invalid when dumping disk or object store PVs. To maintain a consistent output format, the family ID output column is included when dumping disk or object store PVs, but the value is always zero.

--select-locked-volumes

-l

This option limits the output to volumes with a VV condition that restricts their use. In dump_sspvs, EOM is not considered a restrictive VV condition.

Volumes in RW, RO or DOWN condition are listed because use of these volumes is restricted in some fashion. New storage segments may not be created on volumes in RW, RO or DOWN condition. Volumes in RO condition may not be read and volumes in DOWN condition may not be read or written.

When this option is specified, volumes in RW, RO and DOWN condition are listed. Volumes in RWC and EOM condition are not listed.

--select-migration-output

-M

Select only tape volumes that are being written (migration targets). A tape is selected by this option when it is not at EOM but has been written. Tapes that have never been written and tapes that are at EOM are not selected. This option is invalid when dumping disk or object store volumes.

--select-storage-class=*sc*

-c *sc*

Output will be restricted to volumes in the given storage class.

--show-aggregate-stats

-a

Show the statistics associated with aggregated small files on tape. This option is invalid when dumping disk or object store volumes.

--show-priority

-P

Show the current VV priority.

--show-pvr-id

-r

Show the PVR ID of each tape volume. This option is invalid when dumping disk or object store volumes.

--show-pvr-name

-R

Show the PVR Name of each tape volume. This option is invalid when dumping disk or object store volumes.

--show-map-state

-n

Show the storage map state instead of the VV condition in the output.

--show-creation

-C

Show the volume creation time in the output.

--subsystem-id=*ss*

-i *ss*

The number of the subsystem that contains the Core Server of interest. This argument must be a positive non-zero value. If not specified on the command line, the value of the HPSS_SUBSYS_ID environment variable will be used.

--json

-j

Display the output in JSON format.

--vv-name

-v

Provide a virtual volume name to process. This name can include one or more wild card characters, asterisk (*) and question mark (?), for multiple or single character substitution respectively. Only VV names that match the VV name string passed in will be queried and reported.

F.9.5. Repack Output Volumes

Starting in HPSS 7.1, tape volumes containing data written to them by repack will be indicated by the presence of the string "Repack output tape" in the "Other information" field. Tapes will be indicated in this way only if the repack segregation feature is on (it is on by default).

F.9.6. RAIT Tape Volumes

Starting in HPSS 7.x, RAIT tape volumes are denoted with "Rait" in the "Other information" field. "Read-verify" will be added if the volume has the read verify attribute. "Mount-min-PVs" will be added if the volume has that attribute. "Degraded" will be added if the PV has been marked "not mountable", or if it has had a read or write error, or has been marked as not readable or writable by the administrator.

F.9.7. Examples

The following commands will list the tape physical volumes managed by the Core Server in subsystem 1. In this example, Tape0200 is retired.

```
$ export LC_ALL=en_US
$ dump_sspvs --subsystem-id=1

          VV             Storage
Volume    Cond    Family   Class                Space Used \
        Max Volume Size  Other information
Tape0100  EOM          0       3         21,474,836,480,000 \
```

```
              21,474,836,480,000
    Tape0200  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000  Retired
    Tape0300  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape0400  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape0500  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape0600  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape0700  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape0800  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape0900  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape1000  EOM          0       3         21,474,836,480,000 \
              21,474,836,480,000
    Tape1100  RWC          0       3                          0 \
              21,474,836,480,000
                                    ------------------------- \
 --------------------------
    Totals for all tape volumes:            214,748,364,800,000 \
     18,967,330,422,906,283

    Storage Class    7 totals:              214,748,364,800,000 \
     18,967,330,422,906,283
```

This one selects disk volumes in subsystem 1, and takes the environmental defaults (which usually will be the desired values) for -g, -p and -k. Note that Disk0200 is retired and Disk0500 is Offline:

```
    $ export LC_ALL=da_DK
    $ dump_sspvs --subsystem-id=1 --select-disk

              VV             Storage
    Volume    Cond    Family   Class                 Space Used             Max Volume
 Size  Other information
    Disk0000   RO          0      10        115,534,553,153,536
115,534,553,153,536
    Disk0100   RO          0      10        115,534,553,153,536
115,534,553,153,536
    Disk0200   RW          0      10        115,534,553,153,536
115,534,553,153,536
    Disk0300   RO          0      10        115,534,553,153,536
115,534,553,153,536  Retired
    Disk0400   RO          0      10        115,534,553,153,536
115,534,553,153,536
    Disk0500   RO          0      10        115,534,553,153,536
115,534,553,153,536
```

```
    Disk0600   RO          0      10        115,534,553,153,536
115,534,553,153,536
    Disk0700   RO          0      10        115,534,553,153,536
115,534,553,153,536
    Disk0800   RO          0      10        115,534,553,153,536
115,534,553,153,536
    Disk0900   RO          0      10        115,534,553,153,536
115,534,553,153,536
    Disk1000  RWC          0      10        115,534,553,153,536
115,534,553,153,536
                                    -------------------------
-------------------------
    Totals for all disk volumes:          1,270,880,084,689,896
1,270,880,084,689,896

    Storage Class   10 totals:            1,270,880,084,689,896
1,270,880,084,689,896
```

This one lists tape volumes using the "lstapes-format" output option:

```
    $ dump_sspvs --subsystem-id=1 --lstapes-format

    Volume:VVID:SeqNum           SC   Family       Last Write   \
           Last Read     State Segments      Used    Total
    RB0100:000186a4-04-00000001-01ed9ce600cf0176-0d98:0     3        0 2024/11/26
10:01:09\
     ---------- -------- ALLOC      386 3903719.3 20480000.0
    RB0101:000186a4-04-00000001-01ed9ce600d2fe20-0d98:0     3        0 2024/11/26
09:59:21\
     ---------- -------- ALLOC     2812 14639657.9 20480000.0
    RB0102:000186a4-04-00000001-01ed9ce600d42f2a-0d98:0     3        0 ----------
--------\
     ---------- -------- ALLOC        0      0.0 20480000.0
    RB0103:000186a4-04-00000001-01ed9ce600d56188-0d98:0     3        0 2023/06/24
09:10:32\
     2024/11/20 14:50:38 EOM       1678 17543570.1 31557014.2
    RB0104:000186a4-04-00000001-01ed9ce600d68414-0d98:0     3        0 2023/06/25
15:09:46\
     2024/09/30 15:06:06 EOM       1969 30374116.2 31896448.3
    RB0105:000186a4-04-00000001-01ed9ce600d7aaec-0d98:0     3        0 ----------
--------\
     ---------- -------- ALLOC        0      0.0 20480000.0
    RB0106:000186a4-04-00000001-01ed9ce600d9246c-0d98:0     3        0 ----------
--------\
     ---------- -------- ALLOC        0      0.0 20480000.0
    RB0107:000186a4-04-00000001-01ed9ce600da3bcc-0d98:0     3        0 2023/06/24
14:20:03\
     2024/09/30 15:06:35 EOM       1302 36146669.5 44480531.8
    RB0108:000186a4-04-00000001-01ed9ce600db562e-0d98:0     3        0 ----------
--------\
```

```
     ---------- -------- ALLOC        0      0.0 20480000.0
    RB0109:000186a4-04-00000001-01ed9ce600dca858-0d98:0     3        0 2023/07/19
07:59:56\
     2024/09/30 15:06:49 EOM       4490 13888018.2 24172608.6
    RB0110:000186a4-04-00000001-01ed9ce600ddb2fc-0d98:0     3        0 ----------
--------\
     ---------- -------- ALLOC        0      0.0 20480000.0
```

F.9.8. See Also

dumpbf, dumppv_pvr, dumppv_pvl, lsvol

F.10. ff_merge(7)

F.10.1. Name

ff_merge - merge all files and tapes from the source family ID into the destination family ID

F.10.2. Synopsis

ff_merge [--help]
--source-family-id *SOURCE_FAMILY_ID*
--destination-family-id *DESTINATION_FAMILY_ID*
[--subsys-db-name *subsys1,subsys2,subsys3*]
[--schema *SCHEMA*]
[--cfg-db-name *CFG_DB_NAME*]
[--lockfile-path *LOCKFILE_PATH*]

F.10.3. Description

This tool merges all the files and tapes in the specified source file family into the destination file family. It modifies the metadata in the database in the following order: bitfiles → migration records → filesets → tape VVs. Finally, this tool modifies the file family table in the config database.

The metadata changes are commited when updates to ALL affected tables are finished. If there is an error during any update, the metadata transaction will roll back — cancelling all changes — and the program will exit.

Once all files in the source file family are fully merged into the destination file family, this tool removes the source file family from the list of available HPSS file families (from the config database).

F.10.4. Notes

The file family merge is a ONE-WAY operation. Once complete, there is no undo or roll back option. Please ensure that you want to complete the file family merge operation prior to executing it.

Note that this is a metadata-only operation. No HPSS data transfers occur as part of this tool's operation.

The HPSS system must be stopped. This tool will check for a running Core Server and will not proceed if it finds one. Though HPSS must be stopped, the Db2 database must continue running.

F.10.5. Required Arguments

-s,--source-family-id *SOURCE_FAMILY_ID*

Source file family. This family will no longer exist after this program completes.

-d,--destination-family-id *DESTINATION_FAMILY_ID*

Destination file family. After this program completes, all files from the source file family will end up in the specified destination file family.

F.10.6. Optional Arguments

-h, --help

show a help message and exit

--subsys-db-name *subsys1,subsys2,subsys3*

a comma-separated list of subsys database names (default: *subsys1*)

--schema *SCHEMA*

database schema name (default: *hpss*)

--cfg-db-name *CFG_DB_NAME*

config database name (default: *cfg*)

--lockfile-path *LOCKFILE_PATH*

path of lock file that ensures only one instance of this program runs at a time (default: */var/hpss/tmp*)

F.10.7. Prerequisites

The following software prerequisites are required for the ff_merge tool:

- Python 3.6 or later
- the ibm-db Python package (available via *pip install ibm_db* or here: https://pypi.org/project/ibm-db/)
- the HPSS Client API Python package

F.11. hashcheck(7)

F.11.1. Name

hashcheck - HPSS File Hash Interrogation Tool

F.11.2. Syntax

hashcheck [*options*] [*file pathname*]

F.11.3. Description

hashcheck is a utility which can be used to interrogate the HPSS system for file hash digest information. The HPSS system provides the capability that allows a hash digest to be associated with a stored file, using an client API function. The provided digest will be regenerated when the file is copied to off-line storage. The client can also request that a hash digest be generated by the system when the file is copied to off-line storage. The generation of a hash digest can also be triggered by associating a hash algorithm with an HPSS Class of Service (COS).

The primary purpose for the utility is interrogate the HPSS migration records for entries that indicate that migration failures due to a hash digest verification. The utility runs in mode when it is executed without the -S option and without a <file pathname> argument. If the utility encounters migrations that have failed due to verification, information from the migration record and associated file hash descriptor is outputted.

The utility can also be used to query hash digest information of a specific file. The utility runs in this mode when a *file pathname* command line argument is supplied. The default output for this mode includes information from any file hash descriptor found for the file. The -r option can be specified in this mode to also output migration record information associated with the file if it has failed migration due to a hash digest verification failure.

The final mode of operation utility is accessed by running it with the -S command line option. In this mode the hash digest table is scanned for records that have digest set by the user and have not yet been successfully verified. Because the file hash entries are scanned for specific flag setting, this mode will result in high database overhead. Care should be taken before deciding to run the utility in this mode. Please consult your support representative before using this mode of operation. Like the previous mode the, -r option can be specified to output information from any migration record associated with the hash digest record.

F.11.4. Options

--flag=*name*,-f *name*

-f *name*

Scan flags (*USER*, *VERIFIED* & *NONE*).

Used to specify the hash digest flags to look for when running a scan of hash digest information. This option is compatible with the --scan option. This option can be specified multiple times and the supported name values are *USER*, *VERIFIED* and *NONE*. The default flags for the --scan option are

USER and not *VERIFIED*.

--migrinfo,-r

This option is used to request that migration record information be outputted for hash digest records that indicate that they have not been successfully verified. This option is compatible with the --scan option or when specifying a file pathname.

--xml,-x

Used to direct the application to produce XML formatted output. With this option and a style sheet HTML can be produced that contain results from runs of the utility.

--output-file=*name*,-o *name*

Used to direct the output to the specified file name. The caller must have permission to create or update the specified file name.

--scan,-S

The utility should scan the hash digest information looking for records that have digest set by the user but have not yet been successfully verified. Because running in this mode can impact performance, please consult your support representative before running the utility in this mode.

--authentication-mechanism=*am*,-m *am*

The security mechanism used to authenticate the client. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--authenticator-name=*an*,-k *an*

The authenticator which is used to obtain credentials. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--db-schema-name=*sn*,-s *sn*

The ASCII name of the Db2 schema that contains the global database. If not specified on the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--global-db-name=*dn*,-g *dn*

The ASCII name of the global database where the basic configuration information for HPSS is found. If not specified on the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--login-principal=*lp*,-p *lp*

The server principal name which this utility will run under. Lsvol is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided on the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--subsystem-id=*ss*,-i *ss*

The number of the sub-system that contains the Core Server of interest. This argument must be a positive non-zero value. If not provided on the command line, the value of the environment variable HPSS_SUBSYS_ID will be used.

F.11.5. Examples

List digests that have not been successfully verified

```
     $ /opt/hpss/bin/hashcheck -i 1
     Bitfile ID
x'0002f7063cf711e3bffe00145e740d5ad8cf8c34e32e11e2ad5d000000000004'
     Fileset Name        FilesetRoot.3971
     File Name           ./fhash-test/test2.data
     Hash Record:
     Algorithm           md5
     Flags               USER
     Creator             HPSSSUM
     Timestamp           10/24/13 16:55:31
     Digest              78D70E87A21E02A52E39691F9B4B62EC
     Migration Record[0]:
     Storage Class       1
     Hierarchy           5
     Family              0
     Create Time         10/24/13 16:55:31
     Verify Failures     154
     Migration Failures  0

     Bitfile ID
x'31a9690045a011e3ba8e00145e740d5ad8cf8c34e32e11e2ad5d000000000004'
     Fileset Name        FilesetRoot.3971
     File Name           ./tmp/file_null_char_digest
     Hash Record:
     Algorithm           md5
     Flags               USER
     Creator             GRIDFTP
     Timestamp           11/04/13 16:26:49
     Digest              983C41ADFA4382DB0086C069262F8588
     Migration Record[0]:
     Storage Class       1
     Hierarchy           5
     Family              0
     Create Time         11/04/13 16:26:49
     Verify Failures     1
     Migration Failures  0
```

Notice that the first file hasn't been verified because verification of the hash digest has failed 154. The second file hash failed verification only once.

Get the hash for a specific file

```
     $ /opt/hpss/bin/hashcheck -i 1 /tmp/gucTests/workload/file2
     Bitfile ID
x'a15de08e434611e3b0e300145e740d5ad8cf8c34e32e11e2ad5d000000000004'
```

```
     Fileset Name        FilesetRoot.3971
     File Name           ./tmp/gucTests/workload/file2
     Hash Record:
     Algorithm           md5
     Flags               USER
     Creator             HPSSSUM
     Timestamp           11/01/13 17:40:38
     Digest              A98C50C68724C33583977CF183775DD0

     $ /opt/hpss/bin/hashcheck -i 1 -r /tmp/gucTests/workload/file2
     Bitfile ID
x'a15de08e434611e3b0e300145e740d5ad8cf8c34e32e11e2ad5d000000000004'
     Fileset Name        FilesetRoot.3971
     File Name           ./tmp/gucTests/workload/file2
     Hash Record:
     Algorithm           md5
     Flags               USER
     Creator             HPSSSUM
     Timestamp           11/01/13 17:40:38
     Digest              A98C50C68724C33583977CF183775DD0
     Migration Record[0]:
     Storage Class       1
     Hierarchy           1
     Family              0
     Create Time         11/01/13 17:40:38
     Verify Failures     0
     Migration Failures  0
```

Notice that the -r option was supplied on the second run and as a result the associated migration record information is also displayed.

Get the hash info for all files

```
     $ /opt/hpss/bin/hashcheck -i 1 -S -r
     Bitfile ID
x'0002f7063cf711e3bffe00145e740d5ad8cf8c34e32e11e2ad5d000000000004'
     Fileset Name        FilesetRoot.3971
     File Name           ./fhash-test/test2.data
     Hash Record:
     Algorithm           md5
     Flags               USER
     Creator             HPSSSUM
     Timestamp           10/24/13 16:55:31
     Digest              78D70E87A21E02A52E39691F9B4B62EC
     Migration Record[0]:
     Storage Class       1
     Hierarchy           5
     Family              0
     Create Time         10/24/13 16:55:31
     Verify Failures     154
```

```
     Migration Failures  0

     ...

     Bitfile ID
x'a16e45b4434611e3b0e300145e740d5ad8cf8c34e32e11e2ad5d000000000004'
     Fileset Name        FilesetRoot.3971
     File Name           ./tmp/gucTests/workload/file3
     Hash Record:
     Algorithm           md5
     Flags               USER
     Creator             HPSSSUM
     Timestamp           11/01/13 17:40:38
     Digest              B9E6B3E6F753A6932188945D6703BD4E
     Migration Record[0]:
     Storage Class       1
     Hierarchy           1
     Family              0
     Create Time         11/01/13 17:40:38
     Verify Failures     0
     Migration Failures  0

     Bitfile ID
x'db88b0fc401911e387ea00145e740d5ad8cf8c34e32e11e2ad5d000000000004'
     Fileset Name        FilesetRoot.3971
     File Name           ./tmp/my_guc_tests/file1
     Hash Record:
     Algorithm           md5
     Flags               USER
     Creator             HPSSSUM
     Timestamp           10/31/13 16:50:33
     Digest              E5E0130A30F03558542580DA0A106C6B
     Migration Record[0]:
     Storage Class       1
     Hierarchy           1
     Family              0
     Create Time         10/31/13 16:50:33
     Verify Failures     0
     Migration Failures  0
```

F.11.6. See Also

hpsssum

F.12. hpss_abort_io(7)

F.12.1. Name

hpss_abort_io - Cancel HPSS I/O Requests

F.12.2. Syntax

hpss_abort_io -h [-a *auth mechanism*] [-k [-t *keytab*]] [-l] [-p *principal*] options

F.12.2.1. Options for each abort type:

[-i *subsystem*] [-v] *REQID* ...

[-i *subsystem*] [-v] -s

[-i *subsystem*] [-v] -u *username*

[-i *subsystem*] [-v] -m *moverid*

[-i *subsystem*] [-v] -n *movername*

F.12.3. Description

This program cancels HPSS I/O requests. There are two modes of basic execution for this tool. The tool can either take a list of request ID strings or cancel all I/O requests associated with the authenticated user for the given subsystem.

For empowered users, the tool can also cancel I/O by Mover Descriptive Name, Mover ID, or for an arbitrary user.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

The tool also can take a list of space or newline separated request IDs from stdin via pipes or file redirects.

hpss_abort_io also allows for processing of partial request ID strings (a part of a request ID at least 4 characters in length). Note that this method may be slower for large batch processing than providing the complete request ID. In the case where a partial request ID would match multiple requests, an error will be displayed and the request will not be aborted.

F.12.4. Options

-h, --help

Display usage information.

--authtype *type*

Authentication type

--keytab

Use keytab for auth

--ktpath *keytab-path*

Set keytab path

--newcred

Create new creds (implies password)

--principal *principal*

Use principal name

-i, --subsys *subsys id*

Subsystem to use (default=1)

-s, --self

Cancel all I/O for the authenticated user.

-v, --verbose

Verbose output

For empowered users only:

-u, --user *user*

Cancel all I/O requests for the given user.

-m,--mover *mover id*

Cancel all I/O requests for the provided Mover.

-n,--mvrname *mvrname*

Cancel all I/O requests for the provided Mover Descriptive Name (requires empowered user).

F.12.5. Example

Example 1: Cancel all active I/O for myuser

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpss_abort_io -s
```

Example 2: Cancel a specific request ID

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpss_abort_io c36108e222680f4cb86008a2a4247cfb cd6703a8fc123741815677186b11a1c6
```

Example 3: Cancel a request using a partial request ID

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpss_abort_io c36108 cd670
```

Example 4: Cancel a specific request ID (from stdin)

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# echo "c36108e222680f4cb86008a2a4247cfb cd6703a8fc123741815677186b11a1c6" |
hpss_abort_io
```

Example 5: (empowered uses only) Cancel all active I/O for a specified user (baduser)

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpss_abort_io -u baduser
```

Example 6: (empowered uses only) Cancel all active I/O for a specific mover (ID 12)

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpss_abort_io -m 12
```

Example 7: (empowered uses only) Cancel all active I/O for a specific mover (using descriptive name)

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpss_abort_io -n "My Disk Mover"
```

Example 8: Cancel a request with user/password authentication

```
# hpss_abort_io -l -s
```

F.13. hpss_acct(7)

F.13.1. Name

hpss_acct - Produce an accounting report

F.13.2. Syntax

hpss_acct [--help]
[--subsystem *subsystem id*]
[--global_db_name *database name*]
[--schema_name *schema name*]
[--ordering *ordering string*]
[--force]

F.13.3. Description

"hpss_acct" reads the HPSS accounting metadata and produces an accounting report for the specified subsystem. The report includes the total amount of data stored, the total number of file accesses, and the total amount of data transferred in and out of the subsystem for each account.

hpss_acct is the same utility used by SSM to produce an accounting report. The utility can be run manually as an alternative to invoking it from the SSM gui.

The accounting policy is used to configure several parameters which control the format of the report and its location. See the "Accounting" section of the HPSS Admin Guide for details of configuring the accounting policy and interpreting the accounting report.

F.13.4. Parameters

-?, --help

Prints out the list of all available arguments.

-i *subsystem id*, --subsystem *subsystem id*

Sets which subsystem for which to generate an accounting report.

-g *database name*,--global_db_name *database name*

Sets the name of the HPSS global database that holds all configuration information for the system.

-s *schema name*,--schema_name *schema name*

Sets the schema name under which the databases tables are located.

-o *ordering string*, --ordering *ordering string*

Accounting may report users based on their Account ID, User ID, Group ID, and/or Realm ID. These are specified by the characters *a*, *u*, *g*, and *r* respectively. This string determines which of these identifiers to use and the order of the characters in the string determines their priority for sorting. For example, "ua" would report users based on their User ID and Account ID. The

report's tables will be sorted by User ID first, followed by Account ID.

-f, --force

Accounting checks the database for other running accounting programs. If another is detected to be running on the specified subsystem, it will stop and exit. This flag forces the accounting program to run anyway. Use with caution! This is mainly used for cases where a previous accounting run was stopped prematurely leaving its status in the database as running. Verify that no other instances of accounting are running before using this flag.

F.13.5. Example

```
$ hpss_acct -i 1
```

F.13.6. Files

/var/hpss/acct

Default directory for accounting reports.

F.14. hpss_dump_acct(7)

F.14.1. Name

hpss_dump_acct - HPSS accounting time series metrics capture

F.14.2. Syntax

hpss_dump_acct [-r]
[-d]
[-p *HPSS bin path*]
[-f *log file*]

F.14.3. Description

hpss_dump_acct is a wrapper script that executes the dump_acct_sum utility with specific options to capture comprehensive accounting information from the HPSS system. The script runs dump_acct_sum with the -b (bandwidth), -c (capacity), -S (summary), and -j (JSON) options to collect detailed accounting data.

This script is typically run from a cron job to periodically capture accounting metrics for monitoring, reporting, and capacity planning purposes. Only one instance should run at a time.

F.14.4. Output Format

@timestamp

ISO 8601 UTC timestamp indicating when the metrics were captured.

msgid

Series identifier for the metric type. This script produces multiple series: **hpss_acct_capacity** (detailed capacity metrics by COS/realm/account/UID/GID), **hpss_acct_bandwidth** (detailed bandwidth metrics by COS), **hpss_cos_capacity** (capacity summary by COS), **hpss_capacity_sum** (overall capacity summary), **hpss_cos_bandwidth** (bandwidth summary by COS), and **hpss_bandwidth_sum** (overall bandwidth summary).

Metric Fields

Specific data fields vary by series type. For example, hpss_bandwidth_sum includes TotalBytesRead, TotalBytesWritten, TotalFilesRead, and TotalFilesWritten.

F.14.5. Parameters

-r

Prepare the output for ELK (Elasticsearch, Logstash, Kibana) and write it to the log file. Without this option, output is written to stdout.

-d

Enable additional debug output. When enabled, the raw dump_acct_sum output is displayed to stdout before processing.

-p *HPSS bin path*

The path to the HPSS bin directory containing dump_acct_sum. Default is **/opt/hpss/bin**.

-f *log file*

The output log file path. Default is **/var/hpss/log/hpss_dump.log**.

F.14.6. Exit Status

0

Success. Accounting information was captured and processed successfully.

1

Failure. An error occurred during execution. Error messages are written to stderr.

F.14.7. Files

/opt/hpss/bin/dump_acct_sum

The HPSS accounting summary utility that provides the raw data.

/var/hpss/log/hpss_dump.log

Default output log file for processed accounting data.

/opt/hpss/lib/py-packages/parsedumpacctsum.py

Python script that parses and formats the dump_acct_sum output.

F.14.8. Information Collected

The script collects the following accounting information:

Capacity Summary

Total number of files and bytes stored per Class of Service (COS), broken down by realm, account, user ID (UID), and group ID (GID).

Bandwidth Summary

Total bytes read and written per COS, including the number of files read and written.

Summary Tables

Overall totals for the entire subsystem, including total files stored, total bytes stored, total bytes read/written, and total files read/written.

F.14.9. Example 1 - Display to stdout

```
$ hpss_dump_acct
```

This displays the accounting summary to standard output without formatting for ELK.

Example output (one JSON object per line):

```
{"@timestamp": "2026-04-20T14:30:02.799Z", "msgid": "hpss_bandwidth_sum", "Subsystem":
1, "TotalBytesRead": 3745893238330, "TotalBytesWritten": 6394586249103,
"TotalFilesRead": 2458436, "TotalFilesWritten": 29473811}
{"@timestamp": "2026-04-20T14:30:02.799Z", "msgid": "hpss_capacity_sum", "Subsystem":
1, "COS": 1, "Realm": "HPSS", "Account": "default", "UID": 1000, "GID": 100,
"TotalFiles": 125000, "TotalBytes": 5500000000000}
{"@timestamp": "2026-04-20T14:30:02.799Z", "msgid": "hpss_summary", "Subsystem": 1,
"TotalFilesStored": 2500000, "TotalBytesStored": 15000000000000, "TotalBytesRead":
3745893238330, "TotalBytesWritten": 6394586249103}
```

F.14.10. Example 2 - Log to file with ELK formatting

```
$ hpss_dump_acct -r
```

This captures the accounting data and appends it to the default log file (/var/hpss/log/hpss_dump.log) in ELK-compatible format.

F.14.11. Example 3 - Debug mode

```
$ hpss_dump_acct -d
```

This runs in debug mode, showing the raw dump_acct_sum output before processing.

F.14.12. Example 4 - Custom output file

```
$ hpss_dump_acct -r -f /var/log/hpss/accounting.log
```

This captures the accounting data and writes it to a custom log file location.

F.14.13. See Also

dump_acct_sum(7), hpss_acct(7), hpss_server_metrics(7), hpss_db_metrics(7)

F.14.14. Notes

- The script uses temporary files in /tmp which are cleaned up on exit.
- Only one instance of this script should run at a time to avoid conflicts.
- The script requires Python to be available for output parsing.
- The accounting data is obtained from the HPSS account capacity and bandwidth summary tables in the database.

F.15. hpss_dump_db_metrics(7)

F.15.1. Name

hpss_dump_db_metrics - HPSS database time series metrics capture

F.15.2. Syntax

hpss_dump_db_metrics [-r]
[-d]
[-p *HPSS bin path*]
[-s *subsystem name*]
[-f *log file*]

F.15.3. Description

hpss_dump_db_metrics is a wrapper script that executes the hpss_db_metrics utility to capture comprehensive database metrics from the HPSS system. The script runs hpss_db_metrics with the -a (all metrics) and -j (JSON output) options to collect detailed database statistics.

This script is typically run from a cron job to periodically capture database metrics for monitoring, performance analysis, and capacity planning purposes. Only one instance should run at a time.

F.15.4. Output Format

@timestamp

ISO 8601 UTC timestamp indicating when the metrics were captured.

msgid

Series identifier for the metric type. This script produces multiple series: **hpss_purge_locked_by_sc** (purge-locked data by storage class), **hpss_purge_locked_summary** (purge-locked summary), **hpss_to_migrate_by_sc** (files to migrate by storage class), **hpss_to_migrate_summary** (migration queue summary), **hpss_to_purge_by_sc** (files to purge by storage class), **hpss_to_purge_summary** (purge queue summary), **hpss_trashcan_by_sc** (trashcan data by storage class), and **hpss_trashcan_summary** (trashcan summary).

Metric Fields

Specific data fields vary by series type. Common fields include Count (number of files), TotalSize (total bytes), User, StorageClass, and COS identifiers.

F.15.5. Parameters

-r

Prepare the output for ELK (Elasticsearch, Logstash, Kibana) and write it to the log file. Without this option, output is written to stdout.

-d

Enable additional debug output. When enabled, the hpss_db_metrics command being executed

is displayed to stdout.

-p *HPSS bin path*

The path to the HPSS bin directory containing hpss_db_metrics. Default is **/opt/hpss/bin**.

-s *subsystem name*

The subsystem name to query. Default is **HSUBSYS1**.

-f *log file*

The output log file path. Default is **/var/hpss/log/hpss_db_metrics.log**.

F.15.6. Exit Status

0

Success. Database metrics were captured and processed successfully.

1

Failure. An error occurred during execution. Error messages are written to stderr.

F.15.7. Files

/opt/hpss/bin/hpss_db_metrics

The HPSS database metrics utility that provides the raw data.

/var/hpss/log/hpss_db_metrics.log

Default output log file for processed database metrics data.

/opt/hpss/lib/py-packages/parsehpssdbmetrics.py

Python script that parses and formats the hpss_db_metrics output.

F.15.8. Metrics Collected

The script collects all available database metrics, including:

Purge-Locked Data

Count and total size of all purge-locked data, grouped by user and storage class.

Trashcan Data

Count and total size of all trashcan data, grouped by user and Class of Service (COS).

Files to Migrate

Count and total size of all files to migrate, grouped by storage class.

Files to Purge

Count and total size of all files to purge, grouped by storage class.

F.15.9. Example 1 - Display to stdout

```
$ hpss_dump_db_metrics
```

This displays the database metrics to standard output without formatting for ELK.

Example output (one JSON object per line):

```
{"@timestamp": "2026-04-20T21:33:54.155Z", "msgid": "hpss_purge_locked_summary",
"bytes": 0, "fileCount": 0}
{"@timestamp": "2026-04-20T21:33:54.155Z", "msgid": "hpss_trashcan_summary", "bytes":
0, "fileCount": 0}
{"@timestamp": "2026-04-20T21:33:54.155Z", "msgid": "hpss_to_migrate_summary",
"bytes": 0, "fileCount": 0}
{"SCLASS_ID": 1, "TOTAL_BYTES": 4418373687, "FILE_COUNT": 1000014, "@timestamp":
"2026-04-20T21:33:54.155Z", "msgid": "hpss_to_purge_by_sc"}
{"@timestamp": "2026-04-20T21:33:54.155Z", "msgid": "hpss_to_purge_summary", "bytes":
4418373687, "fileCount": 1000014}
```

F.15.10. Example 2 - Log to file with ELK formatting

```
$ hpss_dump_db_metrics -r
```

This captures the database metrics and appends them to the default log file (/var/hpss/log/hpss_db_metrics.log) in ELK-compatible format.

F.15.11. Example 3 - Debug mode with custom subsystem

```
$ hpss_dump_db_metrics -d -s HSUBSYS2
```

This runs in debug mode, showing the hpss_db_metrics command being executed, and queries the HSUBSYS2 subsystem.

F.15.12. Example 4 - Custom output file

```
$ hpss_dump_db_metrics -r -f /var/log/hpss/db_metrics.log
```

This captures the database metrics and writes them to a custom log file location.

F.15.13. See Also

hpss_db_metrics(7), hpss_server_metrics(7), hpss_dump_acct(7), hpss_dump_server_metrics(7)

F.15.14. Notes

- The script uses temporary files in /tmp which are cleaned up on exit.
- Only one instance of this script should run at a time to avoid conflicts.
- The script requires Python to be available for output parsing.
- The database metrics are obtained by querying the HPSS database directly.
- The subsystem name parameter allows querying different HPSS subsystems if multiple subsystems are configured.

F.16. hpss_dump_server_metrics(7)

F.16.1. Name

hpss_dump_server_metrics - HPSS server time series metrics capture

F.16.2. Syntax

hpss_dump_server_metrics [-r]
[-d]
[-p *HPSS bin path*]
[-f *log file*]

F.16.3. Description

hpss_dump_server_metrics is a wrapper script that executes the hpss_server_metrics utility to capture comprehensive metrics from all HPSS servers. The script runs hpss_server_metrics with the -t all (all report types) and -j (JSON output) options to collect detailed performance and operational data.

This script is typically run from a cron job to periodically capture server metrics for monitoring, performance analysis, and capacity planning purposes. Only one instance should run at a time.

When the -d (debug) option is not specified, the script automatically includes the -r (reset) option to reset resettable statistics after collection.

F.16.4. Output Format

@timestamp

ISO 8601 UTC timestamp indicating when the metrics were captured.

msgid

Series identifier for the metric type. This script produces multiple series corresponding to different report types: **hpss_server_metrics_mover** (mover statistics), **hpss_server_metrics_storage** (storage summary), **hpss_server_metrics_config** (configuration counts), **hpss_server_metrics_core_limits** (core server limits), **hpss_server_metrics_statfs** (file system statistics by COS), **hpss_server_metrics_core_usage** (core server cache and API statistics), **hpss_server_core_trash_metrics** (trashcan statistics), and **hpss_server_metrics_device_stats** (device and drive I/O statistics).

Metric Fields

Specific data fields vary by series type. For example, hpss_mover includes RequestCount, BytesMoved, and ErrorCount; hpss_drives includes BytesRead, BytesWritten, Mounts, and Errors per device/drive.

F.16.5. Parameters

-r

Prepare the output for ELK (Elasticsearch, Logstash, Kibana) and write it to the log file. Without this option, output is written to stdout.

-d

Enable additional debug output. When enabled, the raw hpss_server_metrics output is displayed to stdout before processing. In debug mode, statistics are NOT reset (the -r flag is not passed to hpss_server_metrics).

-p *HPSS bin path*

The path to the HPSS bin directory containing hpss_server_metrics. Default is **/opt/hpss/bin**.

-f *log file*

The output log file path. Default is **/var/hpss/log/hpss_server_metrics.log**.

F.16.6. Exit Status

0

Success. Server metrics were captured and processed successfully. Note that success is returned even if some servers are unavailable, as long as metrics from available servers were collected.

1

Failure. An error occurred during execution. Error messages are written to stderr.

F.16.7. Files

/opt/hpss/bin/hpss_server_metrics

The HPSS server metrics utility that provides the raw data.

/var/hpss/log/hpss_server_metrics.log

Default output log file for processed server metrics data.

/opt/hpss/lib/py-packages/parsehpssservermetrics.py

Python script that parses and formats the hpss_server_metrics output.

F.16.8. Metrics Collected

The script collects all available server metrics, including:

mover

Number of requests, bytes moved, and errors for each mover.

storage

Summary of all storage on the system.

config

Count of configured HPSS servers and devices.

limits

Limits information for each core server.

statfs

File system statistics for each Class of Service (COS).

core

Core server cache hit rates, API call frequencies, and error rates.

trash

Trash can statistics for each core server.

drives

Bytes read, bytes written, mounts, and errors for each device and drive.

F.16.9. Handling Unavailable Servers

When collecting metrics that require connecting to servers, the tool will continue to report metrics for all available servers even if some are unavailable. A warning message is printed to stderr for each unavailable server, but the tool exits with success status (0) and reports all available metrics.

F.16.10. Example 1 - Display to stdout

```
$ hpss_dump_server_metrics
```

This displays the server metrics to standard output without formatting for ELK. Statistics are NOT reset in this mode.

Example output (one JSON object per line, showing different series types):

```
{"MoverName": "Mover (silverwolf)", "success": true, "Id": 8, "RequestTasks": 0,
"ActiveRequests": 0, "RequestProcessed": 10107, "DataTransfers": 4590,
"RequestErrors": 24, "BufferSize": 33554432, "BytesMoved": 5600742792,
"TotalMoveTime": 23448.216656, "LastStatReset": "2026-03-24T00:54:46Z", "@timestamp":
"2026-04-20T21:50:00.151Z", "msgid": "hpss_server_metrics_mover"}
{"TotalDiskVVs": 2, "TotalDiskPVs": 2, "BytesUsedDisk": 5960048640, "BytesFreeDisk":
100676706304, "BytesTotalDisk": 106636754944, "TotalTapeVVs": 9, "TotalTapePVs": 9,
"BytesUsedTape": 4418373687, "BytesFreeTape": 38654088020516, "BytesTotalTape":
115964116992000, "TotalObjStorVVs": 1, "TotalObjStorPVs": 1, "BytesRefObjStor": 0,
"ObjsRefObjStor": 0, "FileCntObjStor": 0, "@timestamp": "2026-04-20T21:50:00.151Z",
"msgid": "hpss_server_metrics_storage"}
{"TotalHPSSServers": 8, "TotalSubsystems": 1, "TotalNodes": 2, "TotalMovers": 1,
"TotalDiskDevices": 2, "TotalTapeDevices": 4, "TotalTapeMounts": 1691,
"TotalObjectStoreDevices": 1, "@timestamp": "2026-04-20T21:50:00.151Z", "msgid":
"hpss_server_metrics_config"}
{"COSID": 1, "success": true, "Blocksize": 2048, "Blockcount": 56670973950,
"Freeblockcount": 44959770, "Freeblockavailable": 44959770, "FilesystemId": 1,
"Filemaxname": 256, "Filesystemname": "silverwolf", "msgid":
```

```
"hpss_server_metrics_statfs"}
{"SubsystemId": 1, "success": true, "MaxOpenBitfiles": 410000, "CurOpenBitfiles": 0,
"MaxActiveIO": 4980, "CurActiveIO": 0, "MaxActiveCopyIO": 1195, "CurActiveCopyIO": 0,
"MaxConnections": 5000, "CurConnections": 3, "TapeQueueMethod": "Offset",
"TapeQueueActiveSet": 1024, "AllowAnyCharacters": "true", "TrashcanEnabled": "false",
"TrashcanUnlinkEligible": 864000, "XMLSizeLimit": 131072, "XMLRequestLimit": 204800,
"UDASearchObjectLimit": 4096, "UDASearchXMLLimit": 131072, "UDASearchXMLObjLimit":
2048, "MaxRQRequestThreads": 20, "MaxRQDiscoveryThreads": 5, "@timestamp": "2026-04-
20T21:50:00.151Z", "msgid": "hpss_server_metrics_core_limits"}
{"Count": 0, "Rate": 0.0, "Errors": 0.0, "@timestamp": "2026-04-20T21:50:00.151Z",
"msgid": "hpss_server_metrics_core_usage", "action": "AbortJobByMoverId", "server_id":
5}
{"Count": 97, "Rate": 4.121172673311243e-05, "@timestamp": "2026-04-20T21:50:00.151Z",
"msgid": "hpss_server_metrics_core_usage", "action": "FoundStateInList", "server_id":
5}
{"Count": 0, "Rate": 0.0, "@timestamp": "2026-04-20T21:50:00.151Z", "msgid":
"hpss_server_metrics_core_usage", "action": "LengthOfStateList", "server_id": 5}
{"Count": 0, "Rate": 0.0, "@timestamp": "2026-04-20T21:50:00.151Z", "msgid":
"hpss_server_metrics_core_usage", "action": "SelectAndReadTheEntries", "server_id": 5}
{"Id": 5, "Name": "Core Server", "success": true, "trashcansEnabled": false,
"trashIncineratorIsCurrentlyBurning": false, "timeOfLastIncineratorBurn": "1970-01-
01T00:00:00Z", "totalSuccessfulDeletes": 0, "totalFailedDeletes": 0,
"totalBytesIncinerated": 0, "successfulDeletesDuringLastBurn": 0,
"failedDeletesDuringLastBurn": 0, "bytesIncineratedDuringLastBurn": 0,
"totalEntriesInTrashTable": 0, "totalTrashTableEntrySize": 0,
"numEligibleTrashTableEntries": 0, "sizeOfEligibileTrashTableEntries": 0,
"@timestamp": "2026-04-20T21:50:00.151Z", "msgid": "hpss_server_core_trash_metrics"}
{"DeviceName": "/opt/disk/demo_disk", "DriveId": 7, "DeviceId": 7,
"MountsSinceLastMaint": 12, "DriveErrorCount": 0, "MoverId": 8, "NumberOfErrors": 0,
"BytesRead": 1400090624, "BytesWritten": 1398157312, "@timestamp": "2026-04-
20T21:50:00.151Z", "msgid": "hpss_server_metrics_device_stats"}
```


F.16.11. Example 2 - Log to file with ELK formatting

```
$ hpss_dump_server_metrics -r
```

This captures the server metrics and appends them to the default log file (/var/hpss/log/hpss_server_metrics.log) in ELK-compatible format. Resettable statistics are reset after collection.

F.16.12. Example 3 - Debug mode

```
$ hpss_dump_server_metrics -d
```

This runs in debug mode, showing the raw hpss_server_metrics output before processing. Statistics are NOT reset in debug mode.

F.16.13. Example 4 - Custom output file

```
$ hpss_dump_server_metrics -r -f /var/log/hpss/metrics.log
```

This captures the server metrics and writes them to a custom log file location.

F.16.14. See Also

hpss_server_metrics(7), hpss_db_metrics(7), hpss_dump_acct(7), hpssadm_servers_list(7)

F.16.15. Notes

- The script uses temporary files in /tmp which are cleaned up on exit.
- Only one instance of this script should run at a time to avoid conflicts.
- The script requires Python to be available for output parsing.
- In normal operation (without -d), resettable statistics are automatically reset after collection. This includes core server statistics, mover statistics, and device/drive statistics.
- Device statistics for "Mounts since last maintenance" and "Drive error count" are not reset.
- The tool will continue to collect metrics from available servers even if some servers are unavailable, and will exit with success status.

F.17. hpssadm_devices_and_drives_list(7)

F.17.1. Name

hpssadm_devices_and_drives_list - HPSS devices & drives time series metrics capture

F.17.2. Syntax

hpssadm_devices_and_drives_list [-r]
[-d]
[-U *user*]
[-k *keytab*]
[-p *HPSS bin path*]
[-f *log file*]

F.17.3. Description

hpssadm_devices_and_drives_list is a wrapper script that connects to the HPSS System Manager via hpssadm and captures the output of the "list -Devices and Drives" command. This command displays information about all configured devices and drives in the HPSS system, including their status, type, and operational state.

This script is typically run from a cron job to periodically capture device and drive status for monitoring purposes. Only one instance should run at a time.

F.17.4. Output Format

@timestamp

ISO 8601 UTC timestamp indicating when the metrics were captured.

msgid

Series identifier, set to **hpss_devices_and_drives** for all device and drive records.

Metric Fields

Device and drive information including DeviceName, DriveName, DeviceType, Status, State, Host, and other operational attributes specific to each device and drive.

F.17.5. Parameters

-r

Prepare the output for ELK (Elasticsearch, Logstash, Kibana) and write it to the log file. Without this option, output is written to stdout.

-d

Enable additional debug output. When enabled, the raw hpssadm output is displayed to stdout before processing.

-U *user*

The hpssadm user name to use for authentication. Default is **ssmuser**.

-k *keytab*

The path to the keytab file for the hpssadm user. Default is **/var/hpss/etc/ssmuser.keytab**.

-p *HPSS bin path*

The path to the HPSS bin directory containing hpssadm.pl. Default is **/opt/hpss/bin**.

-f *log file*

The output log file path. Default is **/var/hpss/log/hpssadm_dnd.log**.

F.17.6. Exit Status

0

Success. Device and drive list information was captured and processed successfully.

1

Failure. An error occurred during execution. Error messages are written to stderr.

F.17.7. Files

/opt/hpss/bin/hpssadm.pl

The HPSS command-line System Manager client.

/var/hpss/etc/ssmuser.keytab

Default keytab file for SSM authentication.

/var/hpss/log/hpssadm_dnd.log

Default output log file for processed device and drive data.

/opt/hpss/lib/py-packages/parsednd.py

Python script that parses and formats the hpssadm output.

F.17.8. Example 1 - Display to stdout

```
$ hpssadm_devices_and_drives_list
```

This displays the devices and drives list to standard output without formatting for ELK.

Example output (one JSON object per line):

```
{"ID": "1", "Device Type": "Generic - Default Disk", "Device Name":
"/opt/disk/hpss_disk", "Device State": "Enabled", "Device Admin State": "Unlocked",
"Mover": "Mover (silverwolf)", "Mover Host": "silverwolf.hpsslab.ibm.com", "Drive
Address": "", "Drive Flag": "Clear", "Drive State": "Enabled", "Drive Admin State":
"Unlocked", "Comment": "", "PVR": "", "Mounted Volume": "HY000100", "State":
```

```
"Enabled", "Library Unit ID": "", "Capacity": 96636764160, "msgid":
"hpss_devices_and_drives", "@timestamp": "2026-04-20T22:19:36.210Z"}
{"ID": "2", "Device Type": "IBM - 3580 Gen8 Tape", "Device Name":
"/dev/hpss/mhvtl_lto-8_04", "Device State": "Enabled", "Device Admin State":
"Unlocked", "Mover": "Mover (silverwolf)", "Mover Host": "silverwolf.hpsslab.ibm.com",
"Drive Address": "T10:IBM|ULT3580-TD8     XYZZY_A4", "Drive Flag": "Clear", "Drive
State": "Enabled", "Drive Admin State": "Unlocked", "Comment": "", "PVR": "SCSI PVR",
"Mounted Volume": "", "State": "Enabled", "Library Unit ID": "-1", "Capacity": 0,
"msgid": "hpss_devices_and_drives", "@timestamp": "2026-04-20T22:19:36.210Z"}
{"ID": "3", "Device Type": "IBM - 3580 Gen8 Tape", "Device Name":
"/dev/hpss/mhvtl_lto-8_03", "Device State": "Enabled", "Device Admin State":
"Unlocked", "Mover": "Mover (silverwolf)", "Mover Host": "silverwolf.hpsslab.ibm.com",
"Drive Address": "T10:IBM|ULT3580-TD8     XYZZY_A3", "Drive Flag": "Clear", "Drive
State": "Enabled", "Drive Admin State": "Unlocked", "Comment": "", "PVR": "SCSI PVR",
"Mounted Volume": "", "State": "Enabled", "Library Unit ID": "-1", "Capacity": 0,
"msgid": "hpss_devices_and_drives", "@timestamp": "2026-04-20T22:19:36.210Z"}
{"ID": "4", "Device Type": "IBM - 3580 Gen8 Tape", "Device Name":
"/dev/hpss/mhvtl_lto-8_02", "Device State": "Enabled", "Device Admin State":
"Unlocked", "Mover": "Mover (silverwolf)", "Mover Host": "silverwolf.hpsslab.ibm.com",
"Drive Address": "T10:IBM|ULT3580-TD8     XYZZY_A2", "Drive Flag": "Clear", "Drive
State": "Enabled", "Drive Admin State": "Unlocked", "Comment": "", "PVR": "SCSI PVR",
"Mounted Volume": "", "State": "Enabled", "Library Unit ID": "-1", "Capacity": 0,
"msgid": "hpss_devices_and_drives", "@timestamp": "2026-04-20T22:19:36.210Z"}
{"ID": "5", "Device Type": "IBM - 3580 Gen8 Tape", "Device Name":
"/dev/hpss/mhvtl_lto-8_01", "Device State": "Enabled", "Device Admin State":
"Unlocked", "Mover": "Mover (silverwolf)", "Mover Host": "silverwolf.hpsslab.ibm.com",
"Drive Address": "T10:IBM|ULT3580-TD8     XYZZY_A1", "Drive Flag": "Clear", "Drive
State": "Enabled", "Drive Admin State": "Unlocked", "Comment": "", "PVR": "SCSI PVR",
"Mounted Volume": "", "State": "Enabled", "Library Unit ID": "-1", "Capacity": 0,
"msgid": "hpss_devices_and_drives", "@timestamp": "2026-04-20T22:19:36.210Z"}
{"ID": "6", "Device Type": "AWS - S3 Object Store", "Device Name":
"http://s3.silverwolf.hpsslab.ibm.com", "Device State": "Enabled", "Device Admin
State": "Unlocked", "Mover": "Mover (silverwolf)", "Mover Host":
"silverwolf.hpsslab.ibm.com", "Drive Address": "", "Drive Flag": "Clear", "Drive
State": "Enabled", "Drive Admin State": "Unlocked", "Comment": "", "PVR": "", "Mounted
Volume": "00000600", "State": "Enabled", "Library Unit ID": "-1", "Capacity": 0,
"msgid": "hpss_devices_and_drives", "@timestamp": "2026-04-20T22:19:36.210Z"}
```


F.17.9. Example 2 - Log to file with ELK formatting

```
$ hpssadm_devices_and_drives_list -r
```

This captures the device and drive list and appends it to the default log file (/var/hpss/log/hpssadm_dnd.log) in ELK-compatible format.

F.17.10. Example 3 - Debug mode with custom keytab

```
$ hpssadm_devices_and_drives_list -d -k /var/hpss/etc/custom.keytab
```

This runs in debug mode, showing the raw hpssadm output, using a custom keytab file.

F.17.11. Example 4 - Custom output file

```
$ hpssadm_devices_and_drives_list -r -f /var/log/hpss/devices.log
```

This captures the device list and writes it to a custom log file location.

F.17.12. See Also

hpssadm(7), hpss_server_metrics(7), hpssadm_hs_info(7)

F.17.13. Notes

- This script should be run by a user with appropriate HPSS SSM permissions.
- The script uses temporary files in /tmp which are cleaned up on exit.
- Only one instance of this script should run at a time to avoid conflicts.
- The script requires Python to be available for output parsing.

F.18. hpssadm_hs_info(7)

F.18.1. Name

hpssadm_hs_info - HPSS health & status time series metrics capture

F.18.2. Syntax

hpssadm_hs_info [-r]
[-d]
[-U *user*]
[-k *keytab*]
[-p *HPSS bin path*]
[-f *log file*]

F.18.3. Description

hpssadm_hs_info is a wrapper script that connects to the HPSS System Manager via hpssadm and captures the output of the "health_status info" command. This command displays the overall server, device, and space threshold status of the HPSS system.

This script is typically run from a cron job to periodically capture system health metrics for monitoring purposes. Only one instance should run at a time.

F.18.4. Output Format

msgid

Series identifier, set to **hpss_health_status** for all health status records.

Metric Fields

Health status information including ServerStatus (overall server health), DeviceStatus (device health), SpaceThresholdStatus (storage space threshold status), and detailed status information for individual components.

F.18.5. Parameters

-r

Prepare the output for ELK (Elasticsearch, Logstash, Kibana) and write it to the log file. Without this option, output is written to stdout.

-d

Enable additional debug output. When enabled, the raw hpssadm output is displayed to stdout before processing.

-U *user*

The hpssadm user name to use for authentication. Default is **ssmuser**.

-k *keytab*

The path to the keytab file for the hpssadm user. Default is **/var/hpss/etc/ssmuser.keytab**.

-p *HPSS bin path*

The path to the HPSS bin directory containing hpssadm.pl. Default is **/opt/hpss/bin**.

-f *log file*

The output log file path. Default is **/var/hpss/log/hpssadm_hs.log**.

F.18.6. Exit Status

0

Success. Health status information was captured and processed successfully.

1

Failure. An error occurred during execution. Error messages are written to stderr.

F.18.7. Files

/opt/hpss/bin/hpssadm.pl

The HPSS command-line System Manager client.

/var/hpss/etc/ssmuser.keytab

Default keytab file for SSM authentication.

/var/hpss/log/hpssadm_hs.log

Default output log file for processed health status data.

/opt/hpss/lib/py-packages/parsehealthstatus.py

Python script that parses and formats the hpssadm output.

F.18.8. Example 1 - Display to stdout

```
$ hpssadm_hs_info
```

This displays the health status information to standard output without formatting for ELK.

Example output (one JSON object per line):

```
{"msgid": "hpss_hs", "hpss_hs_bytes_moved": "5,600,742,792", "hpss_hs_bytes_used":
"10,378,422,327", "hpss_hs_data_transfers": "4,590", "hpss_hs_pvl_jobs": "2",
"hpss_hs_server_status": "Enabled", "hpss_hs_dnd_status": "Enabled",
"hpss_hs_sc_threshold": "Critical", "hpss_hs_pvr_cart_threshold": "OK"}
```

F.18.9. Example 2 - Log to file with ELK formatting

```
$ hpssadm_hs_info -r
```

This captures the health status and appends it to the default log file (/var/hpss/log/hpssadm_hs.log) in ELK-compatible format.

F.18.10. Example 3 - Debug mode with custom user

```
$ hpssadm_hs_info -d -U admin -k /var/hpss/etc/admin.keytab
```

This runs in debug mode, showing the raw hpssadm output, using the *admin* user with a custom keytab file.

F.18.11. See Also

hpssadm(7), hpss_server_metrics(7)

F.18.12. Notes

- This script should be run by a user with appropriate HPSS SSM permissions.
- The script uses temporary files in /tmp which are cleaned up on exit.
- Only one instance of this script should run at a time to avoid conflicts.
- The script requires Python to be available for output parsing.

F.19. hpssadm_sc_list(7)

F.19.1. Name

hpssadm_sc_list - HPSS active storage class time series metrics capture

F.19.2. Syntax

hpssadm_sc_list [-r]
[-d]
[-U *user*]
[-k *keytab*]
[-p *HPSS bin path*]
[-f *log file*]

F.19.3. Description

hpssadm_sc_list is a wrapper script that connects to the HPSS System Manager via hpssadm and captures the output of the "list -Active Storage Classes" command. This command displays information about all active storage classes in the HPSS system, including their configuration, usage statistics, and current state.

This script is typically run from a cron job to periodically capture storage class information for monitoring and capacity planning purposes. Only one instance should run at a time.

F.19.4. Output Format

@timestamp

ISO 8601 UTC timestamp indicating when the metrics were captured.

msgid

Series identifier, set to **hpss_storage_classes** for all storage class records.

Metric Fields

Storage class information including StorageClassName, StorageClassID, MinFileSize, MaxFileSize, OptimumFileSize, State, and usage statistics such as FilesStored, BytesStored, and capacity information.

F.19.5. Parameters

-r

Prepare the output for ELK (Elasticsearch, Logstash, Kibana) and write it to the log file. Without this option, output is written to stdout.

-d

Enable additional debug output. When enabled, the raw hpssadm output is displayed to stdout before processing.

-U *user*

The hpssadm user name to use for authentication. Default is **ssmuser**.

-k *keytab*

The path to the keytab file for the hpssadm user. Default is **/var/hpss/etc/ssmuser.keytab**.

-p *HPSS bin path*

The path to the HPSS bin directory containing hpssadm.pl. Default is **/opt/hpss/bin**.

-f *log file*

The output log file path. Default is **/var/hpss/log/hpssadm_sc.log**.

F.19.6. Exit Status

0

Success. Active storage classes list was captured and processed successfully.

1

Failure. An error occurred during execution. Error messages are written to stderr.

F.19.7. Files

/opt/hpss/bin/hpssadm.pl

The HPSS command-line System Manager client.

/var/hpss/etc/ssmuser.keytab

Default keytab file for SSM authentication.

/var/hpss/log/hpssadm_sc.log

Default output log file for processed storage class data.

/opt/hpss/lib/py-packages/parseactivesc.py

Python script that parses and formats the hpssadm output.

F.19.8. Example 1 - Display to stdout

```
$ hpssadm_sc_list
```

This displays the active storage classes list to standard output without formatting for ELK.

Example output (one JSON object per line):

```
{"Subsystem": "Subsystem #1", "ID": "7", "Name": "1wd2", "Type": "Disk", "Thresh":
"OK", "Free Space": 0, "Used Space": 0, "Partially Filled VVs": 0, "Total Space": 0,
"Migr State": "None", "Migr Policy": "None", "Purge State": "None", "Purge Policy":
"None", "msgid": "hpss_sc_active", "@timestamp": "2026-04-20T22:02:41.046Z", "Percent
Used": 0.0, "Percent Free": 0.0}
```

{"Subsystem": "Subsystem #1", "ID": "2", "Name": "1wt", "Type": "Tape", "Thresh": "Critical", "Free Space": 0, "Used Space": 0, "Partially Filled VVs": 3, "Total Space": 0, "Migr State": "None", "Migr Policy": "None", "Purge State": "None", "Purge Policy": "None", "msgid": "hpss_sc_active", "@timestamp": "2026-04-20T22:02:41.046Z", "Percent Used": 1.0, "Percent Free": 0.0}
{"Subsystem": "Subsystem #1", "ID": "1", "Name": "1wd", "Type": "Disk", "Thresh": "OK", "Free Space": 92077555712, "Used Space": 5960105984, "Partially Filled VVs": 0, "Total Space": 98037661696, "Migr State": "Waiting", "Migr Policy": "Migration Policy 1", "Purge State": "Waiting", "Purge Policy": "Purge Policy 1", "msgid": "hpss_sc_active", "@timestamp": "2026-04-20T22:02:41.046Z", "Percent Used": 0.07, "Percent Free": 0.93}

F.19.9. Example 2 - Log to file with ELK formatting

```
$ hpssadm_sc_list -r
```

This captures the storage class list and appends it to the default log file (/var/hpss/log/hpssadm_sc.log) in ELK-compatible format.

F.19.10. Example 3 - Debug mode with custom user

```
$ hpssadm_sc_list -d -U admin -k /var/hpss/etc/admin.keytab
```

This runs in debug mode, showing the raw hpssadm output, using the *admin* user with a custom keytab file.

F.19.11. Example 4 - Custom HPSS bin path

```
$ hpssadm_sc_list -r -p /usr/local/hpss/bin
```

This captures the storage class list using a custom HPSS bin directory path.

F.19.12. See Also

hpssadm(7), hpss_server_metrics(7), hpssadm_hs_info(7), hpssadm_devices_and_drives_list(7)

F.19.13. Notes

- This script should be run by a user with appropriate HPSS SSM permissions.
- The script uses temporary files in /tmp which are cleaned up on exit.
- Only one instance of this script should run at a time to avoid conflicts.
- The script requires Python to be available for output parsing.

F.20. hpssadm_servers_list(7)

F.20.1. Name

hpssadm_servers_list - HPSS server time series metrics capture

F.20.2. Syntax

hpssadm_servers_list [-r]
[-d]
[-U *user*]
[-k *keytab*]
[-p *HPSS bin path*]
[-f *log file*]

F.20.3. Description

hpssadm_servers_list is a wrapper script that connects to the HPSS System Manager via hpssadm and captures the output of the "list -Servers" command. This command displays information about all HPSS servers in the system, including their status, type, host, and operational state.

This script is typically run from a cron job to periodically capture server status information for monitoring purposes. Only one instance should run at a time.

F.20.4. Output Format

@timestamp

ISO 8601 UTC timestamp indicating when the metrics were captured.

msgid

Series identifier, set to **hpss_servers** for all server records.

Metric Fields

Server information including ServerName, ServerType, Host, Status, State, and other operational attributes specific to each server type (Core, Mover, Log, etc.).

F.20.5. Parameters

-r

Prepare the output for ELK (Elasticsearch, Logstash, Kibana) and write it to the log file. Without this option, output is written to stdout.

-d

Enable additional debug output. When enabled, the raw hpssadm output is displayed to stdout before processing.

-U *user*

The hpssadm user name to use for authentication. Default is **ssmuser**.

-k *keytab*

The path to the keytab file for the hpssadm user. Default is **/var/hpss/etc/ssmuser.keytab**.

-p *HPSS bin path*

The path to the HPSS bin directory containing hpssadm.pl. Default is **/opt/hpss/bin**.

-f *log file*

The output log file path. Default is **/var/hpss/log/hpssadm_servers.log**.

F.20.6. Exit Status

0

Success. Server list information was captured and processed successfully.

1

Failure. An error occurred during execution. Error messages are written to stderr.

F.20.7. Files

/opt/hpss/bin/hpssadm.pl

The HPSS command-line System Manager client.

/var/hpss/etc/ssmuser.keytab

Default keytab file for SSM authentication.

/var/hpss/log/hpssadm_servers.log

Default output log file for processed server data.

/opt/hpss/lib/py-packages/parseservers.py

Python script that parses and formats the hpssadm output.

F.20.8. Example 1 - Display to stdout

```
$ hpssadm_servers_list
```

This displays the servers list to standard output without formatting for ELK.

Example output (one JSON object per line):

```
{"ID": "1", "Status": "Connected", "Type": "SSMSM", "Subtype": "", "Subsystem":
"Default", "Op State": "Enabled", "Server Name": "SSM System Manager", "Host":
"silverwolf.hpsslab.ibm.com", "Execute Host": "silverwolf", "UUID": "f1b4febf-635b-
8743-8566-76f1b8fd6311", "msgid": "hpss_servers", "@timestamp": "2026-04-
20T22:03:07.992Z"}
```

F.20.9. Example 2 - Log to file with ELK formatting

```
$ hpssadm_servers_list -r
```

This captures the server list and appends it to the default log file (/var/hpss/log/hpssadm_servers.log) in ELK-compatible format.

F.20.10. Example 3 - Debug mode with custom keytab

```
$ hpssadm_servers_list -d -k /var/hpss/etc/custom.keytab
```

This runs in debug mode, showing the raw hpssadm output, using a custom keytab file.

F.20.11. Example 4 - Custom output file and HPSS bin path

```
$ hpssadm_servers_list -r -p /usr/local/hpss/bin -f /var/log/hpss/servers.log
```

This captures the server list using a custom HPSS bin directory and writes it to a custom log file location.

F.20.12. See Also

hpssadm(7), hpss_server_metrics(7), hpssadm_hs_info(7), hpssadm_devices_and_drives_list(7), hpssadm_sc_list(7)

F.20.13. Notes

- This script should be run by a user with appropriate HPSS SSM permissions.
- The script uses temporary files in /tmp which are cleaned up on exit.
- Only one instance of this script should run at a time to avoid conflicts.
- The script requires Python to be available for output parsing.

F.21. hpssadm(7)

F.21.1. Name

hpssadm - The generic name (without programming language extension) for the Command Line SSM.

hpssadm.pl - Startup script for the Command Line SSM written in Perl.

hpssadm.vbs - Startup script for the Command Line SSM written in Microsoft Visual Basic Scripting language.

F.21.2. Table of Contents

This sadly long man page is organized into these sections:

SYNTAX

Command line options.

DESCRIPTION

General overview of the utility.

ENVIRONMENT

Precedence rules for environment variables, command line options, SSM config file settings.

COMMAND OVERVIEW

Command list and basic rules.

STRUCTURE ORGANIZATION

How to deal with fields and subfields of structures in commands.

COMMON COMMAND OPTIONS

Options used by more than one command.

SPECIFYING MULTIPLE COMMAND OPTIONS

Are they OR'ed or AND'ed?

SPECIFYING FIELDS IN BATCH MODE

How to use -field and -subfield options, required in batch.

SPECIFYING FIELDS IN INTERACTIVE MODE

How to maneuver through fields without -field and -subfield options.

COMMAND SET

All the commands, all their options, what they do.

EXAMPLES

Another way to see the commands and what they do.

FILES

Config and log files.

NOTES

Things to watch out for.

F.21.3. Syntax

hpssadm.[pl|vbs] [-A *"Alarm/Event refresh rate"*] [-C *"login.conf file"*] [-D *"Date format pattern"*] [-G *"Alarm/Event get number"*] [-L *"List refresh rate"*] [-N *"Alarm/Event display size"*] [-P *"hpss.jar file"*] [-S *"Session log"*] [-U *"Username"*] [-W *"Wait time"*] [-a *"Authenticator"*] [-b] [-c *"Country code"*] [-d] [-h *"System Manager host name"*] [-i *"SSM preferences path"*] [-j *"Java binary path"*] [-k *"krb5.conf file"*] [-l *"Language code"*] [-m *"SSM configuration file"*] [-n *"port number"|"program number":"program version number"*] [-p *"Client IP address"*] [-r *"RPC protection level"*] [-s *"Security mechanism"*] [-u *"UNIX Realm"*] [-x] # only supported in the .pl version (not .vbs) [--batch-prompt] [--property *"Java property string"*]

F.21.4. Parameters

Note that the parameters for the **hpssadm** utility are the same as those for the hpssgui utility with the following exceptions:

- The -F, -M, -T, -f, and -g options (not listed above) are supported only by the hpssgui utility, not by **hpssadm**.
- The -U, option is supported only by the **hpssadm** utility, not by hpssgui.
- The -b option is supported by both the hpssgui and **hpssadm** utilities but is interpreted differently by each.

For each of the parameters, the associated environment variable name and configuration file variable name (see option -m below) are listed in the table in the ENVIRONMENT section. More details about the architecture of the environment variables and configuration file variables are described in the following sections.

In the parameter descriptions below, notations like "$HPSS_USER_CFG_PATH" refer to environment variables, not Perl variables in the script.

Environment variables and pathnames are given in Unix format. When run under Windows, appropriate adjustments will be needed in environment variables (eg., %HPSS_USER_CFG_PATH% rather than $HPSS_USER_CFG_PATH).

-A "Alarm/Event refresh rate"

The number of seconds to wait between refreshes of the Alarms and Events list.

The default is 5 seconds.



See more information about performance considerations in the NOTES section below.

-C "login.conf file"

Specifies an alternate HPSS login configuration file.

The default is included in the hpss.jar file and is created from the template file in /opt/hpss/config/templates/login.conf.template.

Please see the /opt/hpss/config/templates/login.conf.template file for a description of the format and options.



The $HPSS_PATH_SSM is also used to locate the ssm configuration file and that it only specifies the directory. If the environment variable is used to locate the file, the file name itself is fixed as "login.conf". To use a different file name, it must be specified on the command line, or set in the configuration file.

There is a known exception to the standard option precedence (see the ENVIRONMENT section below) in that the user environment does not override the configuration file for this parameter.

-D "Date format pattern"

Specifies the format pattern for displaying dates.

Example: "kk:mm:ss dd-MMM-yyyy"

This will produce dates formatted as:

13:22:59 03-Jan-2004

For a complete list of the valid formatting options see the Java web page:

http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html

The default date format is based on the default Java date/time formatter for the user's locale.

-G "Alarm/Event get number"

The maximum number of Alarms and Events to grab with each refresh of the Alarms and Events list.

The default is 500.

-L "List refresh rate"

The number of seconds to wait between refreshes of lists. (Applies to all SSM lists except the Alarms and Events list.)

The default is 5 seconds.



See more information about performance considerations in the NOTES section below.

-N "Alarm/Event display number"

The maximum number of alarms/events to display in the Alarms and Events list.

The default is 2000.

-P "hpss.jar file"

Specifies the location of the hpss.jar file.

Defaults to hpss.jar file in the current directory. If that does not exist then the script will try to use $HPSS_SSM_CLASSPATH.

-S "Session log file"

Specifies a pathname to an **hpssadm** session log file. This is extremely useful for diagnosing problems. This option should be used when using the -d option.

By default, session logging is turned off.

-U "username"

Name of the user.

-W "Wait time"

The number of seconds to wait for a managed object or list update from the System Manager.

The default is 5 seconds.



See more information about performance considerations in the NOTES section below.

-a "Authenticator"

The keytab file containing the user's encrypted keytab. If not specified, the default is a mechanism-dependent default in the user's home directory.

Default: \~/keytab (UNIX) or \~/krb5.keytab (KRB5)

-b

Turn on batch mode preference. No user interactive dialog for commands or field updates are issued by default. However, if the --batch-prompt flag is specified, the `hpssadm>` prompt will be displayed at command completion. Updates must be specified using the -field, -subfield, and -item_number options.

By default, **hpssadm** will exit after any error when in batch mode, but this can be changed by turning off the "Exit On Error" preference. The "Echo Commands" preference is also useful in batch mode as an aid to making script output more readable. See the "preference" command for details.

The default is interactive mode.

-c "Country code"

Specifies the National Language Support (NLS) country code to be used by the application when displaying message text.

The default country code is taken from the system properties defined by Java.

-d

Turn on debug preference. Specify the -S option to see the debug level logs in a session log file.

By default, debug is turned off.

-h "System Manager host name"

Specifies the host name of the node on which the System Manager is running.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.

-i "SSM preferences path"

Specifies the path to search for application preferences and configuration.

The default configuration path is the "hpss-ssm-prefs" directory under the user's home directory path, taken from the system properties defined by Java.

-j "Java binary path"

Specifies the path to the java binary.

If not specified on the command line, in the user's environment, or in the configuration file, the script will look for a java binary in the user's $PATH. If one does not exist, the script will fail.

-k "krb5.conf file"

Specifies an alternate Kerberos configuration file which allows the client to authenticate to an alternate Kerberos realm.

Defaults to krb5.conf in the current directory. If that does not exist then it will try to use $KRB5_CONFIG.

If the security mechanism (command line option "-s") is set to "krb5", then this file is required.

-l "Language code"

Specifies the NLS language code to be used by the application when displaying message text.

The default language code is taken from the system properties defined by Java.

-m "SSM configuration file"

Specifies the configuration file that defines the required variables for the SSM client to run properly. Users can also specify other variables in this file as opposed to specifying them on the command line.

The default file used is ssm.conf in the current directory. If that does not exist then the script will try to use $HPSS_PATH_SSM/ssm.conf.



The $HPSS_PATH_SSM is also used for LOGIN_CONF and that it only specifies the directory. If the environment variable is used to locate the file, the file name itself is fixed as "ssm.conf". To use a different file name, it must be specified on the command line, or set in the configuration file and HPSS_PATH_SSM left

unset (although it doesn't make much sense to set it in the configuration file since this setting's job is to specify the location of the configuration file).

-n "port number" or "program number":"program version number"

Specifies the port number that the System Manager is using to accept incoming connections.

OR

Specifies the ONC RPC program number and version that the System Manager is using to register the port to be used to accept incoming connections.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.



The HPSS_SSM_SERVER_LISTEN_PORT can be set to a specific port upon which the site wants the System Manager to listen for client RPCs. If this is 0, then the port will be chosen by portmapper. Setting this to a non-zero value will give client scripts that need to go through a firewall a static port number that the System Manager is listening on; otherwise, the port will change each time the System Manager is cycled.

-p "Client IP address"

Specifies the IP address of the client machine. This should be used when the client is assigned a NAT address from a VPN server for example.

The default address is taken from the local host setting.

-r "RPC protection level"

Specifies the RPC protection level to use when connecting to the System Manager.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.

-s "Security mechanism"

Specifies the security mechanism to use when connecting to the System Manager.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.

Valid values are "krb5" and "unix".

-u "UNIX Realm"

Specifies the UNIX realm. This is often the uppercase fully qualified hostname of the SSM server.

This parameter is only needed if the security mechanism (command line option "-s") is set to "unix".

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line if the security mechanism is "unix". At install time, mkhpss will insert the appropriate value in the SSM configuration file.

-x

Test mode. Just displays the Java command without actually executing it. This option is only supported in the .pl version (not .vbs).

The default value for test mode is off.

--batch-prompt

Turn on displaying the `hpssadm>` prompt in batch mode.

By default, prompting in batch mode is turned off.

--property "Java property string"

Arbitrary property strings can be passed to the Java program being launched using the --property option. Property strings have the form "-D<property>=<value>". Multiple property strings separated by spaces can be passed to the --property option. For example:

i. --property "-Dhpss.ssm.SMConnections=1 -Dother=different" ...

Properties that can be set in this way include

hpss.ssm.SMConnections=*number*

The number of connections to open to the System Manager.

hpss.ssm.mobjects.RPCLoginContext.trace=*true*|*false*

hpss.ssm.mobjects.RPCInterface.trace=*true*|*false*

hpss.jgss.GSSCredentialKRB5.trace=*true*|*false*

hpss.jgss.GSSMechanismKRB5.trace=*true*|*false*

hpss.jgss.GSSMechanismManager.trace=*true*|*false*

hpss.jgss.GSSMechanismUnix.trace=*true*|*false*

Setting these properties to true sends trace output for the methods of each class (and its inner classes) to stdout. Setting them to false suppresses the output. Trace information can be very helpful for trouble-shooting, however bear in mind that collecting trace information can consume considerable disk space, depending upon the number of classes being traced and the load on the system.

hpss.ssm.mobjects.RPCLoginContext.debug=*true*|*false*

hpss.ssm.mobjects.RPCInterface.debug=*true*|*false*

hpss.jgss.GSSCredentialKRB5.debug=*true*|*false*

hpss.jgss.GSSCredentialKRB5.debug=*true*|*false*

hpss.jgss.GSSMechanismManager.debug=*true*|*false*

hpss.jgss.GSSMechanismUnix.debug=*true*|*false*

Setting these properties to true sends debug output for the methods of each class (and its inner classes) to stdout. Setting them to false suppresses the output. Debug information can be very

helpful for trouble-shooting, however bear in mind that collecting debug information can consume considerable disk space, depending upon the number of classes being debugged and the load on the system.

com.ibm.security.jgss.debug=all

When the client runs on an AIX platform, this property enables all categories of debug output for Java's GSS package. The following table shows the various categories that can be requested:

help

List debug categories

all

Turn on debugging for all categories

off

Turn off debugging completely

app

Application debugging (default)

ctx

Context operations debugging

cred

Credentials (including name) operations

marsh

Marshaling of tokens

mic

MIC operations

prov

Provider operations

qop

QOP operations

unmarsh

Unmarshaling operations

unwrap

Unwrap operations

wrap

Wrap operations

com.ibm.security.krb5.Krb5Debug=all

When the client runs on an AIX platform, this property enables debug output for Java's Kerberos package. The only supported values for this property are "all" and "off".

F.21.5. Description

The Command Line SSM, known as **hpssadm**, allows administrators limited management of an HPSS system without the use of the SSM graphical user interface. It may be executed in batch or interactive mode. Some rudimentary mass configuration support is provided by means of the ability to issue config commands from a batch script.

The **hpssadm** utility provides commands to monitor and control HPSS servers, devices, storage classes, filesets, junctions, PVL jobs, and volumes. It allows the display of alarms and events and of all SSM lists. It supports all configuration operations.

For most commands there are no syntax differences between batch or interactive mode. The syntax differs between batch and interactive modes for commands which involve setting an input field. These include configuration add and update commands as well as the volume import, export, create, delete, and move commands. See the sections "SPECIFYING FIELDS IN INTERACTIVE MODE" and "SPECIFYING FIELDS IN BATCH MODE" for details.

The default startup mode of **hpssadm** is interactive. To start an **hpssadm** session in batch mode, use the -b option. To switch to batch mode after **hpssadm** is initialized, set the "Batch" setting of the Session Preferences to "on". To switch back to operating in interactive mode, set the "Batch" setting to "off". See the "preferences" command in the "COMMAND SET" section for more information on these settings.

Whether in batch or interactive mode, the **hpssadm** utility reads commands from its standard input, so when executing it in batch mode, the commands may be conveniently listed in a file and input redirected from this file. Alternatively, commands may be embedded in a shell script by means of a "here document"; for example:

```
hpssadm <<-EOF
server start -all
server list
device list
EOF
```

The **hpssadm** utility is started by the hpssadm.pl or hpssadm.vbs script. All command line options are optional if running in the default environment. Defaults are assigned based on the SSM configuration file (ssm.conf) and environment variables. Here are some key default assumptions:

java binary location

If the java binary location isn't specified by the -j command line option, $JAVA_BIN environment variable or SSM configuration file setting (JAVA_BIN), then it is assumed that it is in the client's $PATH.

SSM configuration file

The SSM client scripts will look for a file named "ssm.conf" in the current directory. If that does not exist then they will try to use $HPSS_PATH_SSM/ssm.conf (which is generated by mkhpss). If neither file exists, they will use default values.

login configuration file

If the login configuration file isn't specified by the -C command line option or in the SSM configuration file (LOGIN_CONFIG), then the SSM client will use the login.conf file that is contained in the `hpss.jar` file. If the `hpss.jar` file does not contain a login.conf file, the SSM client will look for a file named "login.conf" in the current directory.

All **hpssadm** users must be listed as authorized users in the AUTHZACL table (see the "FILES" section of this man page).

The hpssadm.[pl|vbs] programs are scripts which invoke the Java application launcher (the "java" command) on HPSSADM, the main class for the **hpssadm** utility.

Once initialized, the **hpssadm** utility accepts and executes commands from its standard input. See the "COMMAND OVERVIEW" section of this man page for details.

F.21.6. Environment

The **hpssadm** uses a variety of methods to figure out the correct settings for running in the user's environment. These methods include: command line options, environment variable settings and SSM configuration file settings.

The command line options were discussed above and have the highest precedence (override environment variables and configuration variable settings).

For just the hpssadm.pl script, the environment variables are picked up by the user's environment variable settings. If the hpssadm.pl script is running on the HPSS core server machine, then it will attempt to pickup HPSS environment variable settings by running hpss_set_env -all and set any options represented in that list to the values from hpss_set_env(7), overriding the defaults and not changing the script's environment. See the man page for hpss_set_env(7) for more information about how it picks up HPSS environment variable settings from the user's environment and the system's env.conf file.

The hpssadm.vbs script does not take into account environment variables; for the most part, the user won't have access to hpss_set_env(7) from their Microsoft Windows desktop, so this won't be an issue. However, this is a shortcoming, because it won't attempt to pickup HPSS environment variables that the user may have previously set in their Windows environment.

The mkhpss tool generates $HPSS_PATH_SSM/ssm.conf, the SSM configuration file, on the core server machine. This file must be copied and possibly tailored for client machines that want to communicate with the System Manager. See the FILES section below for more information about the SSM configuration file.

Due to the multitude of supported environments (different client platforms and different security mechanisms), a set of defaults are chosen if not set by the command line, environment variable,

and/or SSM configuration file:

authentication mechanism

Kerberos

This means that when you run the **hpssadm** script without any command line options or any override environment/configuration variables set, it will look for:

java binary location

$PATH

SSM configuration file

./ssm.conf, OR $HPSS_PATH_SSM/ssm.conf

krb5 configuration file

./krb5.conf, OR /etc/krb5.conf

login configuration file

login.conf in the `hpss.jar` file, OR ./login.conf

hpss.jar file

./hpss.jar, OR $HPSS_SSM_CLASSPATH

This means that for the following configurations, no command line options are required:

1) Running the hpssadm.pl on the same machine as your HPSS core server machine using Kerberos authentication.

2) Running the hpssadm.pl or hpssadm.vbs on a desktop that was configured using the steps outlined in the HPSS Admin Guide, is within the firewall of the HPSS core server machine, is using Kerberos authentication, and has the java binary in $PATH.

So, to run the **hpssadm** client on a Linux box doing Kerberos authentication without the java binary in the client's $PATH, you would need to do one of the following:

- set JAVA_BIN in the user's environment, or
- set JAVA_BIN in ssm.conf, or
- use the -j command line option to specify the java binary path

For hpssadm.pl, in general, the precedence rule for options is:

1. command line
2. environment
3. SSM configuration file
4. defaults

However, the handling of the SSM configuration file name for hpssadm.pl has a slightly modified precedence:

1. command line
2. environment
3. ./ssm.conf
4. `hpss_set_env HPSS_PATH_SSM`/ssm.conf.

For hpssadm.vbs, which doesn't process environment nor hpss_set_env(7) values, in general, the precedence rule is:

1. command line
2. SSM configuration file
3. defaults

with a slightly modified precedence for the SSM configuration file name:

1. command line
2. ./ssm.conf

The following table illustrates the relationships between command line options, environment variables and SSM configuration variables.

Command Line Argument	Environment Variable	SSM Config Variable
-A	HPSSSSM_UI_ALARM_RATE	HPSS_SSM_ALARM_RATE
-C	HPSS_PATH_SSM/login.conf	LOGIN_CONFIG
-D	none	HPSS_SSM_DATE_FORMAT
-G	HPSS_SSM_ALARMS_GET	HPSS_SSM_ALARMS_GET
-L	HPSSSSM_UI_LIST_RATE	HPSS_SSM_LIST_RATE
-N	HPSS_SSM_ALARMS_DISPLAY	HPSS_SSM_ALARMS_DISPLAY
-P	HPSS_SSM_CLASSPATH	HPSS_SSM_CLASSPATH
-S	none	LOG_FILE
-U	none	HPSSADM_USER_NAME
-W	HPSSSSM_UI_WAIT_TIME	HPSS_SSM_WAIT_TIME
-a	none	HPSSADM_AUTHENTICATOR
-b	none	none
-c	HPSS_SSM_COUNTRY	HPSS_SSM_CNTRY_CODE
-d	none	HPSS_SSM_DEBUG
-h	HPSSADM_SM_HOST_NAME	HPSS_SSM_SM_HOST_NAME
-i	HPSSADM_USER_CFG_PATH	HPSS_SSM_USER_PREF_PATH

-j	JAVA_BIN	JAVA_BIN
-k	KRB5_CONFIG	KRB5_CONFIG
-l	HPSS_SSM_LANGUAGE	HPSS_SSM_LANG_CODE
-m	HPSS_PATH_SSM/ssm.conf	SSM_CONFIG
-n	HPSSADM_SM_PORT_NUM	HPSS_SSM_SM_PORT_NUM
-p	none	HPSS_SSM_CLIENT_IP
-r	HPSSADM_RPC_PROT_LEVEL	HPSS_SSM_RPC_PROT_LEVEL
-s	HPSS_SSM_SEC_MECH	HPSS_SSM_SEC_MECH
-u	HPSSADM_UNIX_DEFAULT_REALM	HPSS_SSM_UNIX_REALM
-x	none	none
--batch-prompt	none	none
--property	none	none

F.21.7. Command Overview

The general format of all **hpssadm** commands is: <command> <verb> [option] [option argument]

```
Available commands are:
alarm
configuration
device
drive_quota
echo
exit
fileset
health_status
help
list
object
preferences
pvl_job
quit
ssm
server
storage_class
task
volume
```

Keywords are case-sensitive.

If any keyword contains internal white space, it must be quoted with the double quote character (").

Commands may be continued across multiple lines by inserting a backslash (\) before the newline. Any other unquoted use of the backslash is a syntax error.

Any keyword may be abbreviated so long as the abbreviation is long enough to distinguish the keyword from other keywords.

Any input text after a pound sign (#) is treated as a comment and ignored. Blank lines are also treated as comments.

F.21.8. Structure Organization

HPSS structures are organized into fields and subfields. Subfields themselves may have subfields. The **hpssadm** utility displays each subfield indented recursively with respect to the field or subfield to which it belongs. For example:

```
        Field 1                     aaa
        Field 2
          Subfield 1
            Subfield 1.1            rrr
            Subfield 1.2            qqq
          Subfield 2
            Subfield 2.1            ddd
          Subfield 3                eee
          Subfield 4                fff
        Field 3                     ggg
```

The fields of this structure are:

```
    Field 1
    Field 2
    Field 3
```

The subfields of field "Field 2" are:

```
    Subfield 1
    Subfield 2
    Subfield 3
    Subfield 4
```

The subfields of subfield "Subfield 1" of field "Field 2" are:

```
    Subfield 1.1
    Subfield 1.2
```

Multiple subfields at the same level may also be grouped several to a line:

```
Field 1                     aaa
Field 2
  Subfield 1
    Subfield 1.1  rrr     Subfield 1.2  qqq <----------------
  Subfield 2
    Subfield 2.1            ddd
  Subfield 3                eee
  Subfield 4                fff
Field 3                     ggg
```

In general, fields for a structure are displayed and grouped in the same order and pattern as used by the corresponding window of the SSM graphical user interface.

Most fields are modified by specifying the new value to which they should be set.

Some fields may not be assigned any arbitrary value but only one of a set of legal values. For example, the Protection Level subfield of the Security Controls subfield of the BasicConfig field of each server config structure may be set only to one of the values

```
Connect
Packet
Packet Integrity
Packet Privacy
```

Some fields, like certain statistics or timestamps, may not be assigned any arbitrary value by the user but may only be reset. Resettable-only fields which are integer types are reset to 0. Resettable-only fields which are timestamps are reset to the current time.

Some fields, like encryption keys, may be set in two ways: by specifying the desired value or by allowing **hpssadm** to generate a random value.

Flag fields are fields which represent a set of flags. The flags represented by a flag field are listed beneath the field name, indented like subfield names. Flag values may be modified by specifying the new value. If a flag is represented in the structure as "on" or "off", the valid values are "on" and "off"; if it is represented as "yes" or "no", the valid values are "yes" and "no". Some flags are read only and may not be changed by the user. **hpssadm** has no visual means of distinguishing these from writable flags. Some flags are not supported for a particular structure type; the value for such flags is listed as "---".

List fields are modified one list element at a time by specifying the item number of the element. For most lists, the item number of each element is displayed as "Item N", where N is a non-negative integer. For some lists, the item number is one of the list column values. The lists that work this second way are:

1. The list of Allowed Classes of Service inside the storage subsystem configuration, for which the item number is the CosId.
2. The list of subsystems and thresholds inside the subsystem-specific storage class thresholds

configuration, for which the item number is the Subsys ID.

See the sections "SPECIFYING FIELDS IN BATCH MODE" and "SPECIFYING FIELDS IN INTERACTIVE MODE" for a description of how to specify fields, subfields, flags, and list item numbers when creating or modifying structures.

F.21.9. Common Command Options

The following options are used by many commands:

-all

Performs the command on all objects.

-field

Specifies the field of the object to be modified in an add or update command in batch mode. See the section on "SPECIFYING FIELDS IN BATCH MODE" for more details.

-id

Specifies a range of integer object ids on which to perform the operations. For some commands, the range is limited to one object.

Example: -id 1,5-7

This expression specifies that the command be performed for object ids 1, 5, 6, and 7.

-item_number

Specifies the item number of an element in a list of an object to be modified in an add or update command in batch mode. See the section on "SPECIFYING FIELDS IN BATCH MODE" for more details.

-name

Specifies a descriptive name for the object upon which to perform the operation. For servers, this is the descriptive name. For volumes, it is the cartridge or volume id. This option is not valid for other object types.

-sick

Performs the command only on objects in the sick list.

-subfield

Specifies the subfield of the object to be modified in an add or update command in batch mode. See the section on "SPECIFYING FIELDS IN BATCH MODE" for more details.

-type

Specifies the type of object, such as the type of configuration structure for configuration commands or the type of volume for volume commands.

F.21.10. Specifying Multiple Command Options

In most cases, where multiple options for a single command are allowed, they are OR'ed together;

all of the options are applied to the operation. For example,

server info -id 1-5 -name "PVR Operator" -basic -specific

would list both the basic and specific information objects (where they exist) for servers 1, 2, 3, 4, and 5 and for the PVR Operator server.

Some multiple options are AND'ed together; the operation is performed on the items which meet the restrictions of all specified options. In the current release, the -sick and -not_in_use options are the only options which are AND'ed. For example,

device list -id 1-5

would list devices 1, 2, 3, 4, and 5, and

device list -sick

would list all sick devices. But

device list -id 1-5 -sick

would generate a listing only of the devices which were both sick and had device ids between 1 and 5.

F.21.11. Specifying Fields in Batch Mode

In batch mode, fields to be modified for object creation or update are specified with the -field, -subfield, and -item_number options. These options are added to the command line of the basic create or update command, which looks like one of:

config add -type XXX -id NNN

config add -type XXX -name NNN

config update -type XXX -id NNN

config update -type XXX -name NNN

The basic syntax for specifying a field which is a simple integer or String type is

-field FIELDNAME VALUE

where FIELDNAME is the label used for displaying the field and VALUE is an integer or string to be assigned to the field. Both the FIELDNAME and the VALUE must be quoted if they include internal white space.

The FIELDNAME may be abbreviated as long as the abbreviation is long enough to distinguish it from the label of any other field in the object. The VALUE may be abbreviated only if it is a string from a selection list; user-defined string VALUEs may not be abbreviated. For example, the selection list of legal values for the Protection Level subfield of the Security Controls subfield of the BasicConfig field of server configurations is:

```
        Connect
        Packet
        Packet Integrity
        Packet Privacy
```

This field may be set to "Packet Privacy" for server 3, which in this example is a Mover, with the abbreviation:

```
config update -type Mover -id 3 \
      -field BasicConfig \
      -subfield Security \
      -subfield Prot "Packet P"
```

But there is no way to abbreviate the VALUE for a user-defined field like Server Name; it must be spelled out. This command, for example, would change the descriptive name of server 3 to "Core Server Subsystem 2":

```
        config update -type Mover -id 3 \
            -field BasicConfig \
                -subfield Basic \
                    -subfield "Server N" "Core Server Subsystem 2"
```

If the field is not a simple integer or string, but some kind of HPSS structure, then the member of that structure which is to be set is treated as a subfield and is specified with the -subfield option:

```
        -field FIELDNAME -subfield SUBFIELDNAME VALUE
```

The SUBFIELDNAME in this case is the label of the subfield.

If the subfield is itself a nonsimple HPSS structure, the member of the subfield which is to be set is indicated by another -subfield option. As many -subfield options as are necessary to get down to a simple integer or string field must be specified:

```
        -field FIELDNAME \
            -subfield SUBFIELDNAME \
                -subfield SUBFIELDNAME \
                    -subfield SUBFIELDNAME VALUE
```

For example, if the Mover Configuration structure were:

```
        Field 1                     aaa
        Field 2
          Subfield 1
            Subfield 1.1
```

```
        Subfield 1.1.1      bbb
        Subfield 1.1.2      ccc
    Subfield 1.2            qqq
  Subfield 2
    Subfield 2.1            ddd
  Subfield 3                eee
  Subfield 4                fff
Field 3                     ggg
```

then to change field "Subfield 1.1.2" from "ccc" to "xxx" for a mover which is server number 18, specify the command:

```
config update -type Mover \
    -id 18 \
    -field "Field 2" \
        -subfield "Subfield 1" \
            -subfield "Subfield 1.1" \
                -subfield "Subfield 1.1.2" xxx
```

The recursive nature of the -subfield option means that two or more subfields of the same field can't be specified with a single -field option. You couldn't change both "Subfield 3" and "Subfield 4" of the above structure like this:

```
-field "Field 2" \                  # WRONG
    -subfield "Subfield 3" xxx \    # WRONG
    -subfield "Subfield 4" yyy      # WRONG
```

because **hpssadm** would interpret the last line to mean it should look for a field called "Subfield 4" which was a subfield of "Subfield 3". The second "-subfield" in the above command means to go down one more level into the structure, not to find another subfield directly beneath "Field 2".

The way to change both "Subfield 3" and "Subfield 4" is:

```
-field "Field 2" \
    -subfield "Subfield 3" xxx \
-field "Field 2" \
    -subfield "Subfield 4" yyy
```

To set items in a list of a subfield, use the -item_number option. For most lists, each item in the subfield list is identified as "Item N", where N is a non-negative integer. For example:

```
Field 1                     aaa
Field 2
  Item 0
    Subfield 2.1            ddd
    Subfield 2.2            eee
```

```
  Item 1
    Subfield 2.1            fff
    Subfield 2.2            ggg
  Item 2
    Subfield 2.1            hhh
    Subfield 2.2            iii
Field 3                     zzz
```

To change Subfield 2.1 of Item 1 in the list to "xxx", specify:

```
-field "Field 2" \
    -item_number 1 \
        -subfield "Subfield 2.1" xxx
```

For some lists, the item number is just the id number of the item in the list. The lists that work this way are:

1) The list of Allowed Classes of Service inside the storage subsystem configuration.

For example, suppose the configuration for storage subsystem 1 looks like this:

```
    Storage Subsystem Configuration

    BasicStruct
      Subsystem ID                              1
      Default Class of Service          NONE
      Subsystem Name                    Subsystem #1
      Database Name                     lilysb1
      Gatekeeper                        Gatekeeper
      Metadata Space Warning Threshold          0
      Metadata Space Critical Threshold         0
      Metadata Space Monitor Interval           0
      DB Log Monitor Interval                   0
      COS Change Stream Count                   0
    GlobalDefaultCOSId  1
    CosList
      CosId  CosName                        Allowed  IsDefault
          1  Disk 5 Level                  yes      yes
         13  Disk/Tape Mult Cp A           yes      no
         15  Disk/Tape Mult Cp ABCD        yes      no
         22  Tape Simple No Tape Copies A  yes      no
         26  Disk/Tape Aggregate           yes      no
         27  Disk/Tape Simple 3-Level      yes      no
         14  Disk/Tape multiple copies 4   yes      no
```

To change an item from the CosList, specify the CosId with the "-item_number" option. For example, to disallow COS 15 from this subsystem configuration, the correct -field syntax is:

```
config up -type "Storage Subsys" -id 1 \
    -field "CosList" \
        -item_number 15 \
            -subfield Allowed no
```

2) The list of subsystems and thresholds inside the subsystem-specific storage class thresholds configuration.

For example, suppose the subsystem-specific threshold list for storage class 101 looks like this:

```
Thresholds for Storage Class by Subsystem

Storage Class ID                101
Storage Class Type          Disk
Storage Class Name          Disk 101
Default Warning Threshold        80
Default Critical Threshold       90
Storage Subsystem Count           9
Threshold Table
  Subsys ID  Subsys Name     Warning  Critical
          1  Subsystem #1    default   default
          2  Subsystem #2    default   default
          3  Subsystem #3    default   default
          4  Subsystem #4         50        80
          5  Subsystem #5    default   default
          6  Subsystem #6    default   default
          7  Subsystem #7    default   default
          8  freddy               10        20
       1001  Subsystem 1001  default   default
```

To change an item from the Threshold list, specify the Subsys ID with the "-item_number" option. For example, to set a Warning threshold of 40 percent for subsystem 5, the correct -field syntax is:

```
config up -type "Threshold" -id 101 \
    -field "Threshold Table" \
        -item_number 5 \
            -subfield Warning 40
```

Subfields may be arranged in a table of rows and columns with no individual field labels. In this case, the SUBFIELDNAME for each subfield is constructed of the field's row header plus one space plus the column header. For example, if the subfields of field Foo are arranged in a table like this:

```
Foo
           colHdr1     colHdr2
  rowHdr1  SUBFIELD_A  SUBFIELD_B
  rowHdr2  SUBFIELD_C  SUBFIELD_D
```

then subfield SUBFIELD_C of field Foo is specified as:

```
-field Foo -subfield "rowHdr2 colHdr1"
```

To set resettable-only fields in batch mode, specify only the field name with no value:

```
-field FIELDNAME
```

To set random-value fields like encryption keys in batch mode to a user-defined value, specify the field name, the (internationalized version of the) keyword "no", and the desired value:

```
-field FIELDNAME no VALUE
```

To allow **hpssadm** to generate a random value for these fields, specify the fieldname and the (internationalized version of the) keyword "yes", with no value:

```
-field FIELDNAME yes
```

To set flag fields in batch mode, use the -subfield suboption with the flag name and (the internationalized versions of) either "on"/"off" or "yes"/"no" as the flag value:

```
-field FIELDNAME -subfield FLAGNAME [on|off]
```

Any field not specified on an update command will be left at its original value. Any field not specified on an add command will be assigned the default value. See the HPSS Admin Guide for default field values.

F.21.12. Specifying Fields in Interactive Mode

In interactive mode, the -field and -subfield options are not used. The **hpssadm** utility prompts for all information needed to update the structure.

The user should type in the appropriate add or update command, specifying the structure type and, if appropriate, either the -id or -name option to identify the structure to be modified:

```
config add -type XXX -id NNN
config add -type XXX -name NNN
config update -type XXX -id NNN
config update -type XXX -name NNN
```

The **hpssadm** utility then retrieves and displays the default or specified object and prompts the user for the field to be modified, using the "Field to modify" prompt. If the user specifies a field which has one or more subfields, **hpssadm** then displays just that field with all its subfields and

prompts the user for the subfield to be modified. This process continues iteratively until the user specifies a field which has no subfields.

Next, **hpssadm** prompts for the value to which to set the field, updates the field to the value specified by the user, and redisplays the modified structure or subfield. Then it prompts for the next field of the structure or subfield to be modified.

To clear a text field, enter one or more spaces at the "New value: " prompt. The spaces may be quoted with the double quote character (") but is not necessary.

To back up out of a subfield into its parent field, the user should type a "return" key at the "Field to modify" prompt. **hpssadm** will then display the parent field with all of its subfields and will prompt the user for the next field to be modified.

To save or cancel the modifications and stop editing the structure, the user should type the "return" key at each "Field to modify" prompt to back up all the way to the top level parent structure. At this point, when **hpssadm** issues the "Field to modify" prompt, the user should type the "return" key once more. **hpssadm** will then prompt for confirmation as to whether to save the modifications, creating or updating the structure in metadata. If the user chooses to save the modifications, **hpssadm** will submit them, tell the user whether the operation worked or not, and return the user to the **hpssadm** prompt. If the user chooses not to save the modifications, **hpssadm** will ask whether he wants to continue working on the operation or exit back to the **hpssadm** prompt.

If the fields of a subfield are arranged in a table of rows and columns, without any individual field names, the field name which the user should specify at the "Field to modify" prompt is the combination of the field's row header plus one space plus the field's column header. See the example in "SPECIFYING FIELDS IN BATCH MODE."

When the user specifies a restricted value set field in interactive mode, **hpssadm** lists the legal values to which the field may be set and then prompts the user for the value.

When the user specifies a resettable-only field in interactive mode, **hpssadm** does not prompt for the value to which to set the field but just resets it and redisplays the structure or subfield.

When the user specifies a random value field in interactive mode, hpssadm prompts for whether the user wants **hpssadm** to generate the value:

```
        Generate a random value? (yes/no)
```

If the user responds "yes", **hpssadm** sets the field to a new random value and redisplays the structure or subfield. If the user responds "no", **hpssadm** prompts for the desired value.

When the user selects a flag field in interactive mode, **hpssadm** prompts for which flag of the field to modify just as if the flag were a normal subfield. It will then will toggle the value of the specified flag. The user does not specify "on" or "off" as he does in batch mode.

When the user selects a field which consists of a list in interactive mode, **hpssadm** prompts for which item of the list to modify, using the prompt "Which item from this list?". The user should respond with the item number of the desired element from the list. For most lists, the item number

is displayed as "Item N", where N is some unique integer. For other lists, the item number is some field from the list element itself. See "SPECIFYING FIELDS IN BATCH MODE" for a description of these lists and which field to specify for the item number.

Any field not modified during this iterative update process will be left at its original value.

F.21.13. Command Set

alarm

```
        list            Displays the one-line summary, using the same fields
                        and format as the hpssgui Alarms and Events
                        window, of the specified alarms.

                        Options: -count NUMBER

                        The NUMBER supplied with the -count option is how
                        many of the most recent alarms to display.  For
                        example, "alarm list -count 20" will display the
                        20 most recent alarms.

        info            Displays the full alarm fields of the specified alarm.

                        Options: -id

                        Multiple ids may be specified.  The id depends upon
                        the value assigned to the alarm by the System Manager
                        in the most recent "alarm list" output.
```

configuration

```
        add             Creates the specified new configuration entry.

                        Options, batch or interactive mode:
                                 -type, -id, -name
                        Options, batch mode only:
                                 -field, -subfield

        info            Displays the specified configuration entry.

                        Options: -type, -id, -name

        update          Updates the specified configuration entry.

                        Options, batch or interactive mode:
                                 -type, -id, -name, -force
                        Options, batch mode only:
                                 -field, -subfield
```

```
delete          Deletes the specified configuration entry.

                Options: -type, -id, -name

set_default     Defines a default structure for the specified type.
                Subsequent add commands for objects of this type
                will use this default in place of the built-in
                default value.  Valid only for config types "Disk
                Device Configuration" and "Tape Device Configuration".

                Note that once you issue the "set_default", all new
                default device config structures will use the device
                id specified in the default structure.  (Normally,
                new default configs use the next available id from
                the device list.)  So any new "config add" command
                must override the device id to avoid id collisions.

                Options, batch or interactive mode:
                         -type
                Options, batch mode only:
                         -field, -subfield

show_default    Displays the default structure for the specified
                type.  This will be the user-defined default if one
                has been specified with the set_default verb, or
                otherwise the system-defined default.  Valid only
                for config types "Disk Device/Drive Configuration"
                and "Tape Device/Drive Configuration".

                Options: -type

The syntax of add and update config operations varies depending
upon whether *hpssadm* is executing in batch or interactive mode.
In interactive mode the fields of the config structure are displayed
and the user is prompted for which ones to modify and the values
to which to set them.  In batch mode, the fields to be modified
and the values to which to set them must be specified with the
"-field" option.

See the "preferences" command description and the "DESCRIPTION",
"SPECIFYING FIELDS IN INTERACTIVE MODE" and "SPECIFYING FIELDS IN
BATCH MODE" sections for for more information on batch and
interactive modes.

The -name option is valid only for server configuration objects.

The "-type" option is required for all configuration commands.
The following configuration types are supported (the type categories
are listed in the same order as they are presented on the hpssgui
"Configure" menu):
```

```
            Storage Subsystem Configuration

            Policy configuration types:
                Accounting Policy
                Logging Policy
                Disk Migration Policy
                Disk Subsystem Migration Policy
                Tape Migration Policy
                Tape Subsystem Migration Policy
                Purge Policy
                Subsystem Purge Policy

            Storage Space configuration types:
                Disk Storage Class Configuration
                Tape Storage Class Configuration
                Thresholds for Storage Class by Subsystem
                Hierarchy
                Class of Service Configuration

            Server configuration types:
                Core Server Configuration
                Gatekeeper Configuration
                Migration Purge Server Configuration
                Mover Configuration
                PVL Configuration
                AML PVR Configuration
                SCSI PVR Configuration
                STK PVR Configuration
                Operator PVR Configuration
                RAIT Engine Configuration
                SSM System Manager Configuration
                Startup Daemon Configuration

            Global configuration types:
                Global Configuration

            Device and Drive configuration types:
                Disk Device Configuration
                Tape Device Configuration

            File Family Configuration

        The info, add, update, and delete commands are supported for all
        types, except that delete is not accepted for "Thresholds for
        Storage Class by Subsystem".  In a sense, subsystem-specific
        storage class thresholds can't be added or deleted, only updated.
        However, both the "add" and "update" commands are accepted for
        modifying thresholds.  To delete a threshold, use the "add" or
        "update" command to modify the thresholds and specify the literal
        string "default" for the desired thresholds.
```

```
The set_default and show_default commands are supported only for
types "Disk Device Configuration" and "Tape Device Configuration".

The force option on the update command is supported only for Disk
Device Configurations.  It is effective only when the Mover field
is modified and the original mover is not up.  This configuration
change has the potential for the corruption of data, since the
original mover cannot be notified that it no longer controls the
device.  In interactive mode without the force option, *hpssadm* will
explain the risks to the user and prompt for confirmation before
allowing the modification.  In batch mode without the force flag,
the operation will always be denied with an HPSS_EDRV_UPDATE error.
In interactive or batch mode with the force option specified,
*hpssadm* will skip the explanation and the prompt and will silently
allow the mod.  For changes to other fields of the Disk Device
Configuration and to other configuration structures, the force
option is ignored.

Like other keywords, the type names may be abbreviated so long as the
abbreviation is unique.  Any type name or abbreviation which includes
white space must be enclosed in double quotes, but may still be
abbreviated within the double quotes.

For updating, reading, or deleting items, the -id or -name option
must be used to specify the desired server; the -id option must
be used to specify the desired device.
```

device

```
list            Lists the same fields supplied on the default hpssgui
                Devices and Drives window for all the devices in the
                specified range.

                Options: -id, -all, -sick

info            Lists the device or drive info for the specified device.

                Options: -id, -device, -drive

update          Updates the specified device entry.

                Options, batch or interactive mode:
                         -type, -id
                Options, batch mode only:
                         -field, -subfield

                The following types are supported with the -type option:

                Mover Device Information
                PVL Drive Information
```

```
                    Like other keywords, the type names may be abbreviated
                    so long as the abbreviation is unique.  Any type name
                    or abbreviation which includes white space must be
                    enclosed in double quotes, but may still be abbreviated
                    within the double quotes.

        lock        Locks the specified device or drive.

                    Options: -id, -device, -drive

        unlock      Unlocks the specified device or drive.

                    Options: -id, -device, -drive

        repair      Marks the specified device or drive repaired.

                    Options: -id, -device, -drive

        dismount    Attempts to dismount the specified drive.

                    Options: -id

        For device commands, the id required by the -id option is the
        HPSS device id. The -device option requests the command to operate
        on the mover device object and the -drive info option requests it
        to operate on the pvl drive object.

        At least one of the two options "-device" or "-drive" must be
        specified for each verb, with the exception of the "update" and
        "dismount" verbs, and it is permissible to specify both.  For
        the "update" verb, the functionality of the "-device" and "-drive"
        flags is replaced by the "-type" option.   The "dismount" verb
        applies only to the drive managed object and does not require either
        the "-device" or "-drive" option.

        More than one id may be specified with the -id option, except for
        the "update" verb, for which only one id may be specified.
```

drive_quota

```
        list        Lists all drive quota records

        info        Display information about a drive quota

                    Options: -id

        update      Update a drive quota record
```

```
                    Options: -id
```

echo

```
        <no verb>       Echo arguments to standard out.

        This command, which has no verbs or options, just prints out its
        arguments.  Although it may be invoked in either interactive or
        batch mode, its primary use is in batch mode as a way to insert
        comments into the output when *hpssadm* is executed from a script.

        Example:

        *hpssadm* <<-EOF
           echo "The following servers are currently in the Sick list:"
           server list -sick
        EOF
```

exit

```
        (no verb)       Exits the utility, disconnecting from the SSM System
                        Manager.  Synonym for "quit".
```

fileset

```
        list            List all filesets and junctions.

        add             Create a new fileset

                        Options, batch mode only: -field

        info            Display information about a fileset

                        Options: -name

        update          Update a fileset

                        Options, batch or interactive mode: -name
                        Options, batch mode only: -field

        del             Delete a fileset

                        Options: -name

        junc_add        Create a junction pointing to a fileset.

                        Options, batch or interactive mode: -name
```

```
                Options, batch mode only: -field

junc_info       Displays info about a junction and the fileset
                to which it points.

                Options: -junction, -parent_fileset

junc_del        Deletes a junction to a fileset

                Options: -junction, -parent_fileset

The -name option always specifies the name of a fileset.  For
junction create commands, the -name option specifies the name
of the fileset to which the new junction should point.

For junction commands, the -parent_fileset option specifies the
name of the fileset in which the junction exists.  The -junction
option specifies the name of the junction as it appears in the
fileset and junction list.  This is always in the form "./NAME"
and is relative to the root directory of the parent fileset of
the junction.

The fileset add and junc_add commands are each submitted as a
background task.  The "hpssadm> " prompt will return immediately
to the user as soon as the task is submitted.  To see the status
of the operation, use the *hpssadm* "task list" command.  Do not
expect background tasks to be completed in the order in which they
are submitted.
```

health_status

```
info            Displays the overall server, device, and space
                threshold status.
apply_config    Apply pending HPSS configuration changes.
```

help

```
(no verb)       Display help information.

The help facility is very rudimentary and offers nothing beyond
a list of the available commands with their options.

The help command itself has no verbs or options.  Instead, it
accepts as its arguments the keywords for *hpssadm* commands and
verbs.  With no arguments, the help command prints general
information and a list of all available *hpssadm* commands.

The help command assumes a single argument to be the name of
an *hpssadm* command and prints the verbs available for that command.
```

```
        It assumes two arguments to be the names of a command and a valid
        verb for that command, and prints the options available for that
        command/verb combination.
```

list

```
        (no verb)       Display an SSM list.

                        Options: -type

        This command displays any SSM list.  Note that this is a duplication
        of the "list" verb provided by some other commands like "server" and
        "device", which, unlike this "list" command, often provide one or more
        customization options for the list display.  The "list" command just
        displays the entire list with no customization or selection
        capabilities.

        List types supported are:

        Alarms and Events
        Classes of Service
        Devices and Drives
        File Families
        Filesets & Junctions List
        Hierarchies
        Logging Policies
        Migration Policies
        Configured Storage Classes
        Active Storage Classes
        Purge Policies
        RTM Summary List
        Servers
        Subsystems
        Tape Mount Requests
        Tape Check-In Requests
```

object

```
        lookup          Display an HPSS object

                        Options:
                            - object_id
                            - path
```

preferences

```
        info            Display the settings for the session preferences.
```

```
update          Modify the settings for the preferences.

                Options, interactive mode:
                         none
                Options, batch mode only:
                         -field

Supported session preferences are:

Debug           When on, *hpssadm* issues debugging information.

                Default off.  Debug can be turned on at startup
                time by specifying the "-d" option to the hpssadm
                script.  This is useful for debugging initialization
                problems.

Verbose         When on, *hpssadm* issues extra informational messages.

                Default off.

Batch           When on, *hpssadm* does not issue prompts for field
                names or values to which to set fields, nor does it
                issue confirmations for operations.  In batch mode,
                subfields must be specified with the -field and
                -subfield options to the update command.

                When this setting is off (interactive mode), regular
                prompts are issued, including iterative prompts for
                updating fields of structures.  The -field and
                -subfield options are not supported in interactive
                mode.

                Default off.

Echo Commands   When on, *hpssadm* echoes each input command.  This is
                provided as an aid to documenting the output of batch
                scripts, but it can be used in either batch or
                interactive mode.

Exit On Error   When on, *hpssadm* will exit on any syntax error and
                on any error from an operation.  This is primarily
                useful in batch scripts, but may be set on or off
                in either batch or interactive mode.

                Default off in interactive mode, on in batch mode.
```

pvl_job

```
list            List the fields supplied on the hpssgui PVL Jobs
```

```
                    window for the specified jobs.

                    Options: -id, -not_in_use

                    The not_in_use option requests that only "not in use"
                    jobs be displayed.

        info        List the info for the specified pvl job.

                    Options: -id

        cancel      Cancels the specified pvl job.

                    Options: -id

        For pvl_job commands, the id required by the -id option is the
        pvl job id.
```

quit

```
        (no verb)   Exits the utility, disconnecting from the SSM System
                    Manager.  Synonym for "exit".
```

rtm

```
        list        List the fields supplied on the hpssgui RTM
                    Summary List window.

                    Options: -id

                    If no -id option is supplied, then all entries are
                    displayed.

        info        List the RTM Details for the specified Request ID

                    Options: -id

        cancel      Cancels the specified I/O job

                    Options: -id

        For rtm commands, the id required by the -id option is the Request ID
```

server

```
        list        Lists the same fields available from the hpssgui
                    Servers window for all servers in the specified range.
```

```
                Options: -id, -all, -sick, -name

        info            Lists the server info for the specified server.

                        Options: -id, -all, -name, -basic, -specific

                        The basic option returns the basic server info
                        (the generic server managed object) and the -specific
                        option returns the server specific managed object.

                        At least one of the two options "-basic" or "-specific"
                        must be specified.  It is permissible to specify both.

        start           Starts the specified server(s).

                        Options: -id, -all, -name

        shutdown        Shutdowns the specified server(s).

                        Options: -id, -all, -name

        halt            Halts the specified server(s).

                        Options: -id, -all, -name

        reinitialize    Reinitializes the specified server(s).

                        Options: -id, -all, -name

        connect         Makes SSM try to connect to the specified server(s).

                        Options: -id, -all, -name

        repair          Marks the specified server(s) as repaired.

                        Options: -id, -all, -name

        For server commands, the id required by the -id option is the
        SSM ID.  The name required by the -name option is the descriptive
        name of a server.  More than one id or descriptive name may be
        specified.
```

ssm

```
        info            Lists status and statistics about SSM.

                        Options: -sm, -session

                        The -sm option lists information about the current
```

```
                    instance of SSM System Manager to which *hpssadm* is
                    connected.

                    The -session option lists information about the
                    current *hpssadm* session.

        shutdown    Shuts down the System Manager.

                    Options: -sm

                    The -sm option shuts down the System Manager.
```

storage_class

```
        list        Lists all the fields supplied on the default hpssgui
                    Active Storage Classes window for the specified storage
                    classes.

                    Options: -id, -all, -sick, -subsystem

        info        Displays the info for the specified storage class.

                    Options:  -id, -subsystem

        migrate     Forces a migration on the specified storage class.

                    Options:  -id, -subsystem

        purge       Forces a purge on the specified storage class.

                    Options:  -id, -subsystem
```

```
For storage_class commands, the id required by the -id option is
the storage class id.  The -subsystem option is used to specify a
subsystem id.  If no subsystem id is specified, the command will
be interpreted for all specified storage classes.  If a subsystem
is specified, the command will be interpreted for all specified
storage classes as they are used by that subsystem.  Only one
-subsystem option may be specified per command, but multiple
storage class ids may be specified for any command.
```

task

```
        list        Lists the status information for all the tasks
                    submitted from this *hpssadm* session.

        A task is a request which, because it could take a long time to
```

```
complete, is executed by *hpssadm* in a background thread.  Requests
which are executed as background tasks are:

   - the volume import_disk, import_tape, export, create_disk,
     create_tape, delete, and move commands

   - the fileset add and junc_add commands
```

volume

```
create_disk     Creates core server disk resources.

                Options, batch mode only:
                         -field, -subfield

create_tape     Creates core server tape resources.

                Options, batch mode only:
                         -field, -subfield

delete          Deletes core server disk or tape resources.

                Options, batch mode only:
                         -field, -subfield

export          Exports PVL disk or tape volumes.

                Options, batch mode only:
                         -field, -subfield

import_disk     Imports PVL disk volumes.

                Options, batch mode only:
                         -field, -subfield

import_tape     Imports PVL tape volumes.

                Options, batch mode only:
                         -field, -subfield

info            Displays information for the specified volume.

                Options: -type, -name, -object_id

update          Updates the specified volume.

                Options, batch or interactive mode:
                         -type, -name, -object_id
                Options, batch mode only:
                         -field, -subfield
```

```
move            Moves the specified volume.

                Options, batch mode only:
                        -field, -subfield
```

The syntax of all of the volume commands except "info" varies depending upon whether **hpssadm** is executing in batch or interactive mode.

For the update command, in interactive mode the fields of the volume are displayed and the user is prompted for which ones to modify and the values to which to set them. In batch mode, the fields to be modified and the values to which to set them must be specified with the "-field" option.

For the import, export, create, delete, and move commands, in interactive mode the input parameters for the operation are displayed and the user is prompted for which ones to modify and the values to which to set them. In batch mode, the parameters to be modified and the values to which to set them must be specified with the "-field" option.

See the "preferences" command description and the "DESCRIPTION", "SPECIFYING FIELDS IN INTERACTIVE MODE" and "SPECIFYING FIELDS IN BATCH MODE" sections for for more information on batch and interactive modes.

The import, export, create, delete, and move commands each allow the user to build the list of target volumes using the Fill Increment, Fill Count, and Volume Label parameters, much like the corresponding hpssgui windows for these operations. There is no need to specify each volume individually unless the volume names are not in a consecutive range. See the EXAMPLES section.

The import, export, create, delete, and move commands are each submitted as a background task. The "hpssadm> " prompt will return immediately to the user as soon as the task is submitted. To see the status of the operation, use the **hpssadm** "task list" command. Do not expect background tasks to be completed in the order in which they are submitted.

The valid types for volume commands include:

```
        PVL Volume Information
        PVR Cartridge Information
        Core Server Disk Volume
        Core Server Tape Volume
```

All volume types may be specified with the -name option. Core server disk and tape volumes may also be specified by one of the following combinations of options:

```
        object_id UUID -server_id UUID
        object_id UUID -server_name NAME
```

The "-object_id", "-server_id", and "-server_name" options for volumes may not be used alone or in combinations other than the ones above.

For these commands, the name required by the -name option is the 6-character cartridge or volume id. For all but the PVR Cartridge command, the 8-character volume id is also acceptable.

The id required by the -object_id option is the string representation of the object uuid of the volume's core server virtual volume id; for example:

```
0a5ef128-9085-11d4-8982-02608c2d8c88
```

The id required by the -server_id option is the string representation of the server uuid of the volume's core server virtual volume id.

The -server_name option requires the descriptive name of the core server who owns the volume.

F.21.14. Examples

1. List the 2 most recent alarms.

```
hpssadm> alarm list -count 2

The output will wrap across lines:

Alarms and Events

Time Updated by System Manager 14:22:27 12-Nov-2003
Time Received by Client        14:22:29 12-Nov-2003
List
  ID  Log Type  Event Time                Server Name
  33  ALARM     Jan 23, 2006 2:40:40 PM   Migration/Purge Server #4
  34  EVENT     Jan 23, 2006 2:29:14 PM   SSM System Manager

   Routine                  Client PID   Client Node ID Client ID
   mps_UpdateSClassUsage    26553        maryjo
   sm_adm_start_one_server  0            0

   Request ID   Object Class   Severity   Error Code
   0                                       37
   0                          Major       -10028

   MsgID      Text
   MPSR0111   Could not make local connection to Core Server.
   SSMS0137   Startup of server 'Mover and Shaker' initiated
```

1. Get detailed information about alarm number 33 in the list above:

```
hpssadm> alarm info -id 33

Alarm/Event Information
```

```
        ID                                              123
        hdr
          Log Type          ALARM
          Event Time        Jan 23, 2006 2:40:40 PM
          Server Name       Migration/Purge Server #4
          Routine           mps_UpdateSClassUsage
          Client PID                            26553
          Client Node ID    maryjo
          Client ID
          Request ID                               0
          Object Class                            37
          Severity          Major
          Error Code                           -10028
        MsgID MPSR0111
        Text  Could not make local connection to Core Server.
```

1. Add the configuration for a mover named "Mover 1" on host hogwarts (batch mode). Any fields not specified will be assigned the default values.

The example here shows how to set all the fields of the specific config subfield. Note that we specify a "yes" value for the "Encryption Key" field because it is a random field and we want **hpssadm** itself to generate the value. Also, note that we have to specify the index for the Data Hostnames entry to be set:

```
    config add -type "Mover" \
        -field BasicConfig                                      \
            -subfield "Basic Controls"                          \
                -subfield "Server Name"  "Mover 1"              \
        -field "SpecificConfig"                                 \
            -subfield "Encryption Key"     yes                  \
        -field "SpecificConfig"                                 \
            -subfield "Buffer Size"        8192                 \
        -field "SpecificConfig"                                 \
            -subfield "TCP Path Name"      /opt/hpss/bin/hpss_mvr_tcp \
        -field "SpecificConfig"                                 \
            -subfield "Hostname"           hogwarts             \
        -field "SpecificConfig"                                 \
            -subfield "Data Hostnames"                          \
                -subfield                  0      hogwarts_0    \
        -field "SpecificConfig"                                 \
            -subfield "Data Hostnames"                          \
                -subfield                  1      hogwarts_1    \
        -field "SpecificConfig"                                 \
            -subfield "TCP Port"           0                    \
        -field "SpecificConfig"                                 \
            -subfield "Port Range Start"   2001                 \
        -field "SpecificConfig"                                 \
            -subfield "Port Range End"     2099
```

1. Add the configuration for a startup daemon "Startup Daemon 1" on host hogwarts (batch mode). Any fields not specified will be assigned the default values.

   ```
   Field names may be abbreviated so long as they are unique:
   ```

```
config add -type "Startup Daemon"                                \
    -field BasicConfig                                           \
        -subfield "Basic Controls"                               \
            -subfield "Server Name"      "Startup Daemon 1"      \
    -field BasicConfig                                           \
        -subfield "Execution Controls"                           \
            -subfield "Execute H"        hogwarts
```

1. Add a tape drive configuration for a 9840 tape.

```
config add -type "Tape Device"                                   \
    -field "Device ID"                      1001                 \
    -field "Device Type"                    "STK - 9840 Tape"    \
    -field "Mover"                          "Mover 1 (hogwarts)" \
                                                                 \
    -field "Mover Device"                                        \
        -subfield "Basic Information"                            \
            -subfield "Device Name"          /dev/stk_9840_01    \
                                                                 \
    -field "PVL Drive"                                           \
        -subfield "PVR"                      "PVR hogwarts"      \
    -field "PVL Drive"                                           \
        -subfield "Drive Address"            /dev/rmt_9840_01    \
    -field "PVL Drive"                                           \
        -subfield "Controller ID"            1001
```

1. Add a disk drive configuration:

```
config add -type "Disk Device"                                      \
    -field "Device ID"                    1002                      \
    -field "Device Type"                  "Generic - Default Disk" \
    -field "Mover"                        "Mover 2 (hogwarts)"      \
                                                                    \
    -field "Mover Device"                                           \
        -subfield "Basic Information"                               \
            -subfield "Device Name"       /dev/rdisk1               \
    -field "Mover Device"                                           \
        -subfield "Basic Information"                               \
            -subfield "Media Block Size"  8192                      \
    -field "Mover Device"                                           \
        -subfield "Basic Information"                               \
            -subfield "Bytes on Device"   536870912                 \
```

```
                                                                    \
    -field "PVL Drive"                                              \
        -subfield "Drive Address"           /dev/rdisk1             \
    -field "PVL Drive"                                              \
        -subfield "Controller ID"           1001                    \
    -field "PVL Drive"                                              \
        -subfield "Mounted Volume"          rd0001
```

1. Modify the server configuration of "Mover 1" to change its buffer size to 4098, to generate a new Encryption Key for it, and to turn off executability. Turning off executability is done using the "Executable" flag of the "Flags" field. Note that we specify a "yes" value for the "Encryption Key" field because it is a random field and we want **hpssadm** itself to generate the value. Other fields will retain their previous value:

```
config update -type Mover                \
   -name "Mover 1"                       \
                                         \
    -field BasicConfig                   \
        -subfield "Execution Controls"   \
           -subfield Flags               \
               -subfield Executable off  \
                                         \
    -field "SpecificConfig"              \
       -subfield "Encryption Key" yes    \
    -field "SpecificConfig"              \
       -subfield "Buffer Size"    4098
```

1. Modify the configuration of tape drive 5 to change its PVR to "STK PVR 3" (batch mode). All other fields will remain unchanged:

```
 config update -type "Tape Device"   \
    -id 5                            \
    -field "PVL Drive"               \
        -subfield "PVR" "STK PVR 3"
```

1. Modify the configuration of disk drive 2 to turn off the read and write flags. Note that we have to have a separate "-field" option to specify each flag. All other fields will remain unchanged:

```
config update -type "Disk Device"        \
    -id 2                                \
    -field "Mover Device"                \
        -subfield "Device Flags"         \
            -subfield "Read Enabled" off \
    -field "Mover Device"                \
        -subfield "Device Flags"         \
            -subfield "Write Enabled" off
```

1. Delete the mover configuration for "Mover 1". Note that this deletes both the basic and specific configurations:

```
config delete -type Mover \
   -name "Mover 1"
```

1. Display the current default configuration for tape devices. These are the default values used for all tape device creations. If a user-defined default has been specified previously with the "set_default" command, this command displays that user-defined default. Otherwise it displays the system-defined default values.

   ```
   The show_default command is supported only for disk and tape
   configurations:
   ```

```
config show_def -type Tape
```

The output when the system-defined default values are in use will be something like this. The device id, device name, and drive address will be generated based on the last highest device id used:

```
Tape Device Configuration

Device ID    122
Device Type  Generic - Default Tape
Mover        Mover (satchel)
Mover Device
  Basic Information
    Device Name     /dev/device122
  Device Flags
    Read Enabled                on     NO-DELAY Support           on
    Write Enabled               on     Write TM(0) to Sync        off
    Removable Media Support     on     SCSI-2 LBA Positioning     off
    Locate Support              on
PVL Drive
  PVR                Operator PVR
  Drive Address      /dev/drive122
  Controller ID      122
  Polling Interval   60
```

1. Modify the default tape device configuration to use pvr "STK PVR" and the device/drive type to "STK - 9840 Tape". The newly defined default values will be used for all subsequent tape device creations.

The set_default command is supported only for disk and tape configurations:

```
config set_default -type "Tape Device" \
    -field "Device Type" "STK - 9840 Tape" \
    -field "PVL Drive" \
       -subfield "PVR" "STK PVR"
```

The output will show the new default tape device configuration:

```
Device ID    122
Device Type  STK - 9840 Tape
Mover        Mover (satchel)
Mover Device
  Basic Information
    Device Name     /dev/device122
  Device Flags
    Read Enabled                on     NO-DELAY Support           on
    Write Enabled               on     Write TM(0) to Sync        off
    Removable Media Support     on     SCSI-2 LBA Positioning     off
    Locate Support              on
PVL Drive
  PVR                STK PVR
  Drive Address      /dev/drive122
  Controller ID      122
  Polling Interval   60
```

1. Get the device info for device 1:

```
hpssadm> device info -device -id 1

Mover Device Information

Device Metadata
  Basic Information
    Device Name              /dev/raw/raw2
    Device ID                                1
    Device Type              Generic - Default Disk
    Media Block Size                      4096
    Bytes on Device                4,096,000,000
    Starting Offset                          0
    Mover                    Mover (maryjo)
    Administrative State     Unlocked
    Operational State        Enabled
    Usage State              Idle
  Device Flags
    Read Enabled             on      Write Enabled         on
    SAN3P-transfer Enabled   off     Multiple Mover Tasks  on
Device Record
  State
    None
```

```
        Set State
          None
        Volume Flags           0
        Number Of Errors       0
        Block Size          4096
        Volume ID
        Bytes Read             0
        Bytes Written          0
```

1. Turn on debug preference (assumes you are currently in interactive mode):

If you are in interactive mode, you'll be getting prompts:

hpssadm>

Type in the preference update command:

hpssadm> pref update

and **hpssadm** will display the current preferences and prompt you for the field to change:

```
        Session Preferences

        Debug                                   off
        Verbose                                 off
        Batch                                   off
        Echo Commands                           off
        Exit On Error                           off

        Field to modify:

    Type in "Debug" or some abbreviation thereof:

        Field to modify: De

    *hpssadm* will display the possible values for the field and prompt
    for the one you want:

        The values allowed for this field are:
                on
                off

        New value:

    Type in "on":

        New value: on

    *hpssadm* will display the modified structure and another prompt:
```

```
    Session Preferences

    Debug                                   on
    Verbose                                 off
    Batch                                   off
    Echo Commands                           off
    Exit On Error                           off

    Field to modify:

Type "return", and *hpssadm* will display the final version of the
structure and a prompt to do the update:

    Session Preferences

    Debug                                   on
    Verbose                                 off
    Batch                                   off
    Echo Commands                           off
    Exit On Error                           off

    Confirm update? (yes/no)

Type "yes" or an abbreviation thereof:

    Confirm update? (yes/no) y

and *hpssadm* stores the modified structure and gives you a new prompt:

    hpssadm>
```

1. List information about pvl job 7:

```
    hpssadm>  pvl_job info -id 7

PVL Request Information

Job ID            7
Request Type      Async Mount
Request Timestamp 12:35:12 12-Nov-2003
Mounted Volumes   1
Request Status    In Use
Volume List
  Volume     Drive   State     Drive Type
  DORA0100   1       Mounted   Generic - Default Disk
```

1. Display the pvl volume:

```
    hpssadm> vol info -type "PVL Volume" -name hpss01
```

```
PVL Volume Information

Volume ID            HPSS0100
PVR Server
Allocated Client     Core Server
Allocation State     Allocated
Volume Type          Generic - Default Disk
Label Format         HPSS
Operational State    Enabled
Usage State          Busy
Administrative State Unlocked
Cartridge ID         HPSS01
```

1. Display the information for the disk volume using the volume object id and the server uuid:

```
hpssadm>  vol info -type "Core Server Disk Volume" \
   -object_id 34c850ba-3d6c-01d9-8567-00065b8deb25 \
   -server_id eb920e62-3a4c-01d9-8567-00065b8deb25

Core Server Disk Volume

Name                    HPSS0100
VV Condition            RWC          Retired no
More commonly used data
  Number of Active Segments                          12
  Storage Class               1 | Disk Storage Class 1
  Storage Map Flags
  Usable Length                          4,094,689,280
  Free Space                             3,692,036,096
  Stripe Width                                       1
  VV Flags                   Online
  Time Last Read
  Time Last Written
  PVL Job ID                                       150
Less commonly used data
  Actual Length               4,096,000,000
  PV Size                     4,096,000,000
  Cluster Length                  1,048,576
  VV BlockSize                      1048576
  PV BlockSize                         4096
  Creation Time    Nov 23, 2004 11:24:53 AM
  VV ID
    Object ID     34c850ba-3d6c-01d9-8567-00065b8deb25
    Server ID     eb920e62-3a4c-01d9-8567-00065b8deb25
    Sub Type      Disk
    Type          Virtual Volume
  Physical Volumes
    Vol Name   Type                     Flags    Dev ID
    HPSS0100   Generic - Default Disk   Online   1
```

```
        Mvr IP Addr          Mvr              Mvr Host
        160.91.199.85:5001   Mover (maryjo)   maryjo

    Note: you can obtain the object and server uuids for a volume by
    issuing the "vol info" command.
```

1. Using batch mode, create resources on 5 tapes, labeled consecutively X70022 through X70026:

```
    volume create_tape                                          \
        -field "Initialization"                                 \
            -subfield "Core Server"       "Core Server"         \
        -field "Initialization"                                 \
            -subfield "Storage Class"     "10 | Tape Storage Class 10" \
        -field "Initialization"                                 \
            -subfield "VVs"               5                     \
        -field "Assign Physical Volumes"                        \
            -subfield "Fill Count"        5                     \
        -field "Assign Physical Volumes"                        \
            -subfield "Volume Label"      x70022
```

The final output of this command will be:

```
        Item after specified modifications:

        Create Tape Resources

        Initialization Fields - Fill These In First
          Storage Class   10 | Tape Storage Class 10
          Core Server     Core Server
          VVs To Create                          5
        Optional or Informational Fields
          PVs in Each VV                 1
          PV Size           64,424,509,440
        Assign Physical Volumes to Virtual Volumes
          Fill Count                      0
          Fill Increment                  1
          Volume Label            X70022
          Fill Direction          Horizontal
          Total Count                     5
          List of VVs to Create
            Volume 0
            X7002200
            X7002300
            X7002400
            X7002500
            X7002600

    Task submitted.
```

1. Using batch mode, import 10 tapes, labeled consecutively aaa000 through aaa009:

```
volume import_tape                                  \
    -field "Import Information"                     \
        -subfield "Build List of Volumes"           \
            -subfield "Use starting label"          \
                -subfield "Fill Count"         10   \
    -field "Import Information"                     \
        -subfield "Build List of Volumes"           \
            -subfield "Use starting label"          \
                -subfield "Volume Label"       aaa000
```

1. Using batch mode, import a set of tapes which are listed in an external file "/tmp/importListFile":

```
volume import_tape                                    \
    -field "Import Information"                       \
        -subfield "Build List of Volumes"             \
            -subfield "Use list in external file"     \
                -subfield "File"    "/tmp/importListFile"
```

1. Using batch mode, create resources on 5 tapes which are not labeled consecutively into a 3-way stripe storage class:

```
volume create_tape                                      \
    -field "Initialization"                             \
        -subfield "Core Server"            "Core Server"      \
    -field "Initialization"                             \
        -subfield "Storage Class"          "15 | Tape 3-way"  \
    -field "Initialization"                             \
        -subfield "VVs"                    2                  \

    -field "Assign Physical Volumes"                    \
        -subfield "Volume Label"           aaa000             \
    -field "Assign Physical Volumes"                    \
        -subfield "Volume Label"           joe000             \
    -field "Assign Physical Volumes"                    \
        -subfield "Volume Label"           fred00             \
    -field "Assign Physical Volumes"                    \
        -subfield "Volume Label"           moe000             \
    -field "Assign Physical Volumes"                    \
        -subfield "Volume Label"           curly0             \
    -field "Assign Physical Volumes"                    \
        -subfield "Volume Label"           larry0
```

F.21.15. Files

/var/hpss/ssm/ssm.conf

The HPSS SSM configuration file. See the HPSS Admin Guide for details about configuring the HPSS Server Authentication Mechanism(s).

The mkhpss tool generates $HPSS_PATH_SSM/ssm.conf.

Depending on the authentication mechanism (Kerberos or Unix), mkhpss will put slightly different contents in the SSM configuration file. In all cases, the following variables will be defined:

HPSS_SSM_SM_HOST_NAME

host SM is running on

HPSS_SSM_SM_PORT_NUM

port SM is running on

HPSS_SSM_RPC_PROT_LEVEL

RPC protection level used by SM

HPSS_SSM_SEC_MECH

security mechanism used for SM communication

If Unix authentication is enabled, the following variable will also be defined:

HPSS_SSM_UNIX_REALM

fully qualified SM UNIX realm name

For example, if a site had both UNIX and Kerberos authentication configured for the System Manager, the ssm.conf file generated by mkhpss might look like:

```
ssm.conf:
    HPSS_SSM_SM_HOST_NAME   acme
    HPSS_SSM_SM_PORT_NUM    536870913:1
    HPSS_SSM_RPC_PROT_LEVEL connect
    HPSS_SSM_SEC_MECH       unix
    HPSS_SSM_UNIX_REALM     acme.com
```

Users could then select their appropriate authentication mechanism by setting HPSS_SSM_SEC_MECH to *krb5* or *unix*.

A site may personalize each SSM client's SSM configuration file by adding additional variables. For a list of these variables, see the table in the ENVIRONMENT section above. For example, a site may want to define the JAVA_BIN variable (which will be different depending on the client platform). Or perhaps one user wants the HPSS_SSM_DATE_FORMAT to specify the year first whereas another user prefers the month first. All of the SSM configuration settings can be overridden by using command line options, but many users prefer setting everything in the SSM configuration file and

not having to type/pass in any command line options.

The SSM client scripts all abort if they cannot find an SSM configuration file following the precedence rules described in the ENVIRONMENT section above.

A template for this file is provided in: `/opt/hpss/config/templates/ssm.conf.template`

login.conf

The HPSS SSM login configuration file.

The default for this file is stored in the hpss.jar file as login.conf. In most cases, it should not be necessary to override this file.

If there is a need to override options in this file please see the /opt/hpss/config/templates/login.conf.template file and use the -C command line option to specify an alternative file.

A template for this file is provided in: `/opt/hpss/config/templates/login.conf.template`

/etc/krb5.conf

Kerberos V5 configuration file. This file is only required if using Kerberos authentication for SSM. This file is created during the process of configuring Kerberos on the HPSS core server machine.

By default the **hpssadm** script will look for this file in the current directory unless an alternate location is specified (by the -k option) or unless UNIX authentication is specified (by the -m and -u options).



The -u command line option will override the HPSS_SSM_SEC_MECH variable in the SSM configuration file.

A template for this file is provided in: `/opt/hpss/config/templates/krb5.conf.template`

/opt/hpss/bin/hpss.jar

The HPSS SSM client java archive file. This file contains the java binary classes required for starting up SSM clients.

The SSM client scripts all require this file.

AUTHZACL db2 table

The HPSS Authorization ACL table. Every user authorized to use an SSM client must have an appropriate entry in this db2 table. The SSM System Manager will reject connections from users without entries in the table, or with invalid entries in the table (e.g. invalid permissions). The table is read by the System Manager when the user logs in; so whenever SSM users are added or deleted from the table, the System Manager does NOT need to be restarted for the changes to take effect.

Users are added or removed from the table using the hpssuser(7) tool. See the hpssuser(7) man page. Users are added differently depending on the authority granted to the user (admin or operator). See the "Using SSM" section of the HPSS Admin Guide for more information about

SSM authority and the required ACL settings for SSM users in this table.

F.21.16. Notes

When adding a new device/drive configuration, SSM gets a fresh set of defaults if the user changes the Device Type. These defaults include the Device ID and Mover fields. If the Device Type is to be modified from the default, it is necessary to specify it first, and to specify other field values afterwards.

See the hpssadm.masscfg.template for other examples of configuration commands in hpssadm.

hpssadm will not complete administrative commands immediately. These commands depend up on interactions with the HPSS servers and can take several minutes to complete, just as they do when performed from the hpssgui. So, for example, when a "server start" command is issued from an **hpssadm** script, it usually takes several minutes for all the specified servers to start and to initialize, just as it does when the servers are started from the hpssgui. However, the **hpssadm** start command returns immediately, before the startup is complete, in order to allow the user to proceed with additional commands. Therefore, if the next command issued to **hpssadm** is a "server list", "server info", or "server shutdown", it is likely to fail, since the servers are not likely to be up and initialized yet. This is especially noticeable in batch mode.

A similar problem exists when new configurations are added. For example, if a new storage class configuration is added using hpssadm, that storage class will be added to the SSM list of configured storage classes. It will take several seconds for the updated list to be sent from the SM server to the **hpssadm** client. Until the updated list arrives, the new storage class will not be available in the **hpssadm** client for use in, for example, the configuration of a new hierarchy. So if the hpssadm script issues the command to add a new storage class and then immediately tries to create a new hierarchy using that storage class, the hierarchy creation will likely fail with an "invalid input" error. This, too, is especially noticeable in batch mode. Longer polling intervals make the problem worse.

Unfortunately, **hpssadm** does not yet have a graceful way to avoid this. To workaround these problems for now, stalling commands can be inserted into the script. In the above example, the script might issue the storage class create request and then issue several "storage_class list" commands before issuing the hierarchy create request.

hpssadm does not guarantee completion of background tasks in the order in which they are submitted. Do not, for example, issue a volume import command and then immediately issue a volume create_disk or create_tape command for those volumes without first making certain the import completed.

hpssadm can report stale information if the refresh rates (-L and -A options) are too low or the wait time (-W) is too high. The refresh and wait times of **hpssadm** and hpssgui clients can also affect SSM System Manager performance. See the NOTES section of the hpssgui man page for a discussion of the tradeoffs of these values.

F.22. ssmcli(7)

F.22.1. Name

ssmcli - SSM Command Line Interface

F.22.2. Table of Contents

HELP
Command help.

COMMAND STRUCTURE
Command structure and rules.

FILES
Configuration and log files.

DESCRIPTION
General overview of the interface.

CONFIGURATION
Configuration options.

NOTES
Things for consideration.

F.22.3. Syntax

ssmcli [--config-file *"Configuration file"*] [--api-url *"API URL"*] [--log-file *"Log file"*] [--verbose *"Verbose output"*] [--install-completion *"Install auto-completion"*] [--show-completion *"Show auto-completion"*]

F.22.4. Help

You can get help with any command in the **ssmcli**. To do so, add the `--help` or `-h` parameter at the end of a command name.

For example, the following command displays help for the general **ssmcli** options and top-level commands.

```
$ ssmcli --help
```

Another example, the following command displays help for a specific SSM operation.

```
$ ssmcli file-family --help
```

F.22.5. Command Structure

ssmcli + <COMMAND> <SUBCOMMAND> [OPTIONS AND PARAMETERS]

The **ssmcli** uses a multipart structure on the command line that must be specified in this order:

1. The base call to the **ssmcli** program.
2. The top-level command, which corresponds to an SSM operation supported by the **ssmcli**.
3. The subcommand that specifies which operation to perform.
4. Any options or parameters required by the operation. The order does not matter.



Parameters can take various types of trivial and non-trivial input values, such as integers, strings, lists, and JSON structures. Options can vary depending on the defaults of the operation. It's important to consult the help parameter of each command to understand its available parameters, options, and correct usage before running it.

F.22.6. Files

config.yml

The **ssmcli** configuration file. See the HPSS Admin Guide for details on configuring the **ssmcli** with the `configure` command.

ssmcli.log

The **ssmcli** log file output.

F.22.7. Description

The SSM Command Line Interface (CLI), or **ssmcli**, provides a powerful yet easy-to-use command-line tool for controlling and managing an HPSS system.

Unlike the legacy **hpssadm**, the **ssmcli** is designed as a non-interactive command-line interface. Each command performs a single operation and returns the result directly to standard output.

F.22.8. Configuration

Command Line Argument	Environment Variable	Config Variable
--config-file	SSM_CLI_CONFIG_FILE	config_file
--api-url	SSM_CLI_API_URL	api_url
None	None	access_key
--output	SSM_CLI_OUTPUT	output

--log -file	SSM_CLI_LOG_PATH	log_file
None	None	http.verify_ssl
None	SSM_CLI_HTTP_PROXY	http.http_proxy
None	None	http.https_proxy
None	None	http.headers
None	None	http.timeout_connect
None	None	http.timeout_read
None	None	http.timeout_total
None	None	http.allow_redirects
None	None	http.max_redirects
None	None	http.stream
None	None	http.chunk_size
None	None	http.decode_unicode
None	None	http.keep_alive
None	None	http.accept_encoding
None	None	http.user_agent

F.22.9. Notes

All commands in the **ssmcli** correspond to requests made to the SSM API's REST interface API. Each SSM operation with an API endpoint has an API reference that can be found at the `/docs` page.

F.23. hpss_avaledit(7)

F.23.1. Name

hpss_avaledit - HPSS Account Validation Metadata Editor

F.23.2. Syntax

hpss_avaledit [-g *global_db_name*] [-s *schema name*] *action*

F.23.3. Description

The "hpss_avaledit" HPSS utility manages user account validation metadata for HPSS sites that use Site-style accounting where Account Validation has been enabled on the Accounting Policy screen in SSM. You must be authenticated with appropriate access to the metadata file before using this utility.

F.23.4. Options

-g *Global DB Name*

The name of the global database. If not present, the value for this parameter will be taken from the environment variable HPSS_GLOBAL_DB_NAME.

-s *Schema Name*

The schema name associated with the HPSS Databases. If not present, the value for this parameter will be taken from the environment variable HPSS_MM_SCHEMA_NAME.

An action may be one of the following:

```
-add           <user>  [<acct>] [-def]
-default       <user>   <acct>
-list [i|d|a] [<user>] [<acct>]
-remove        <user>  {<acct> | -all}
```

These are described in detail below.

user specifies a single user by uid or moniker and an optional realm identifier. It may be one of the following:

```
-user {<moniker> | <uid>}
-uid   <uid>
-uname <moniker>
```

A numeric value assigned to "-user" is considered to be a uid and a moniker otherwise. A moniker may be a simple name, or fully qualified in the form: <moniker>@<realmname>. Specify a non-local realm uid with: -realm <realm-id-#>

acct specifies an account index by number or name. It may be one of the following.

```
-aname <acct-name-string>
-aidx  <acct-index-number>
```

Actions:

```
-add         <user>  [<acct>] [-def]
```

Create a single user account and print the result. If either an account name or index is supplied, that information is used when creating the account. The user's moniker is used as the account name if none is supplied. The next available account index is used if none is supplied. If the "-def" option is specified, this will be made the user's default account. Even if the "-def" option is not used, if the user has no other account index in the metadata file, this index will become the user's default account index.

```
-default     <user>   <acct>
```

Make the specified account the user's default account index and print the result.

```
-list [i|d|a] [<user>] [<acct>]
```

List the account indexes for one or more users or accounts. If "i" is specified, numeric identifiers and realm IDs are printed instead of monikers. If "d" is specified, all of the fields for each metadata record are dumped in a multiple line format. If "a" is specified, monikers are shown and this is the same as not specifyig a single letter value after the "-list" option. If a user is specified, only accounts for that user are listed. If an account is specified, only indexes for that account are listed. If no users or no accounts are specified, all accounts in the metadata file are printed.

```
-remove      <user>  {<acct> | -all}
```

Remove one or all account indexes for the specified user and print the account indexes removed. The specified account index is removed for the user unless the "-all" option is supplied where all of the user's account indexes are removed. If the user's default account is removed and that user has other account index, it is up to the administrator to chose a new default account index for that user with the "-default" action.

F.23.5. Example

For the following examples, assume there are two users with the following attributes:

```
User                   Realm Id    Uid
joe                     100001     777
```

```
     bob@otherrealm.com       100002     888
```

Add a default account index for user "joe" with an account name of "joe" and the next available account index:

```
% hpss_avaledit -add -user joe -def
```

Add a foreign user to an existing account named "development":

```
% hpss_avaledit -add -user bob@otherrealm.com -aname development
```

Make an existing account index user joe's new default account:

```
% hpss_avaledit -default -user joe -aidx 12345
```

List all accounts for user joe:

```
% hpss_avaledit -list -uname joe
        ACCOUNT  ACCOUNT_NAME    DEFAULT  USER
            277  development     no       joe
           7311  production      no       joe
          12345  joe             yes      joe
```

List all users in the "development" account:

```
     % hpss_avaledit -list -aname development

        ACCOUNT  ACCOUNT_NAME    DEFAULT  USER
            277  development     no       joe
            277  development     no       bob@otherrealm.com
```

List all users in the "development" account as in the previous example, but print out their realm ids and their uids:

```
     % hpss_avaledit -list i -aname development

        ACCOUNT  ACCOUNT_NAME    DEFAULT  USER
            277  development     no       100001:777
            277  development     no       100002:888
```

Dump all users in the "development" account:

```
     % hpss_avaledit -list d -aname development
```

```
  AccountIndex: 277
       RealmId: 100001
        UserId: 777
       GroupId: 0
DefaultAcctIdx: 0x00000000
         Flags: 0
   AccountName: development

  AccountIndex: 277
       RealmId: 100002
        UserId: 888
       GroupId: 0
DefaultAcctIdx: 0x00000000
         Flags: 0
   AccountName: development

Remove all of user joe's account indexes:

% hpss_avaledit -remove -user joe -all

Remove user joe from the development account index:

% hpss_avaledit -remove -uid 777 -aname development
```

F.24. hpsscat(7)

F.24.1. Name

hpsscat - Read and print HPSS files to standard output

F.24.2. Syntax

hpsscat [OPTION] *FILE*...

F.24.3. Description

This program opens one or more HPSS files, reads the data, and writes the data to standard output.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.24.4. Options

--authtype *type*
: Authentication type

--keytab
: Use keytab for auth

--ktpath *keytab-path*
: Set keytab path

--newcred
: Create new creds (implies password)

--principal *principal*
: Use principal name

-b, --buffer-size *bufsize*
: Allows the user to specify the size of the read buffer. Default buffer size is 256Mb.

-n , --no-stage
: Read the file directly from tape if not staged.

-t, --no-tape
: Do not cat the file if it requires staging.

-s, --stats

Print the elapsed data transfer time, not including time to open the file, to stderr.

-h, --help

Display usage information.

F.24.5. Example

Example 1: Cat a file in HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsscat /tmp/path/to/hpss/file
<file contents>
```

Example 2: Cat multiple files in HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsscat /tmp/path/to/hpss/file /tmp/path/to/other/file
<file contents>
<other file contents>
```

F.25. hpsschmod(7)

F.25.1. Name

hpsschmod - Change HPSS file mode bits

F.25.2. Syntax

hpsschmod *MODE FILE...*

F.25.3. Description

This program changes the mode bits for one or more HPSS files. The mode bits must be provided in octal format.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.25.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.25.5. Example

Example 1: Change the permissions of a file

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsschmod 666 /tmp/path/to/hpss/file
```

F.26. hpsschown(7)

F.26.1. Name

hpsschown - Change HPSS file ownership

F.26.2. Syntax

hpsschown *OWNER GROUP FILE...*

F.26.3. Description

This program changes the owner for one or more HPSS files. The UID and GID must be in numeric format, rather than user or group names.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.26.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.26.5. Example

Example 1: Change the owner of an HPSS file to UID/GID 0

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsschown 0 0 /tmp/path/to/hpss/file
```

F.27. HPSS.conf(7)

F.27.1. Name

HPSS.conf - Generic HPSS Configuration File

F.27.2. Description

The HPSS.conf file provides a structured file containing configuration Information for assorted HPSS servers and clients in particular: the HPSS PFTP Client, the HPSS PFTP Daemon, the Transfer Agent and the HPSS Multinode Daemon, and the HPSS movers.

F.27.3. Location

The search path for the HPSS.conf file uses the following algorithm:

1. The path specified by the HPSS_CFG_FILE_PATH environment variable
2. /usr/local/etc
3. /var/hpss/etc

F.27.4. Syntax

The syntax of entries in the HPSS.conf file areas follows:

Level 0 Entries (Stanzas)

One or more of: Flags: A string with no "=" sign Simple Stanzas: A string with a string value after an "=" sign. Compound Stanzas: A string with multiple values (SubStanzas) bracketed by an Equal sign and an opening brace ("= \{") then terminated by a closing brace ("}").

Level 1 Entries (SubStanzas)

One or more of: Flags: A string with No "=" sign Simple SubStanzas: A string with a string value after an "=" sign. Compound SubStanzas: A string with multiple values (Sections) bracketed by an Equal sign and an opening brace ("= \{") then terminated by a closing brace ("}").

Level 2 Entries (Sections)

One or more of: Flags: A string with No "=" sign Simple Sections: A string with a string value after an "=" sign. Compound Sections: A string with multiple values (SubSections) bracketed by an Equal sign and an opening brace, "= \{", then terminated by a closing brace, "}".

Level 3 Entries (SubSections)

One or more of: Flags: A string with No) One or more "=" sign Simple SubSections: A string with a string value after an "=" sign.

Example

```
; A Comment - First character is a ";" or "#" This is a Stanza
Flag, It has NO Value
```

```
This is a Simple Stanza = This is its string value.

Now for a Compound Stanza = {
A SubStanza Flag with NO Value
A Simple SubStanza = Some String Value

Now for a Compound SubStanza = {
   A Section Flag with NO Value
   A Simple Section = Some String Value
   Now for a Compound Section = {
      A Subsection Flag with NO Value
      A Simple Subsection = Some String Value
   }
}
}
```

Limitations

No Line May contain more than 512 characters. Only nine (9) levels are allowed. All Compound Structures MUST be terminated by a closing bracket. All items are case-sensitive. Spaces are generally significant. Applications may specify their own configuration parameters and link with the parsing routines in src/cs/cfg_extension.

HPSS Reserved Level 0 Entries (Stanzas)

"PFTP Client", "PFTP Client Interfaces", "Multinode Table", "Network Options", "PFTP Daemon", "Pipe File", "Transfer Agent", "Local File Path", (Not Implemented but reserved for Future), "PSI" (Not Implemented but reserved for Future)

F.27.5. Examples

See the template file /opt/hpss/config/templates/HPSS.conf.tmpl.

F.27.6. Files

`/opt/hpss/config/templates/HPSS.conf.tmpl`

F.27.7. See Also

hpss_ftp,pftp_client,multinoded,hpss_pftpd

HPSS User's Guide

HPSS Admin Guide

F.27.8. Author

Tim Preston, Sandia National Labs, Albuquerque, NM

F.28. hpsscp(7)

F.28.1. Name

hpsscp - Copy one or more files to or from HPSS

F.28.2. Synopsis

hpsscp [*OPTION*] *SOURCE... DIRECTORY*

F.28.3. Description

This program copies files to or from HPSS. This program supports copying multiple sources to a provided destination directory. By default, the destination is assumed to be local and the source HPSS.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

The client stripe width can be modified by setting **HPSS_API_PIO_STRIPE_WIDTH** in the environment. The client transfer size can be modified by setting **HPSS_API_PIO_BLOCK_SIZE** in the environment. These values can be used to tune the environment for improved transfer performance.

Additionally, the data network interface can be controlled using the **HPSS_API_PIO_USE_API_HOST** environment variable. Setting this will cause the API to use the interface set by **HPSS_API_HOSTNAME** rather than the default hostname interface for the system.

F.28.4. Options

-h, --help

Display usage information.

--authtype *type*

Authentication type

--keytab

Use keytab for auth

--ktpath *keytab-path*

Set keytab path

--newcred

Create new creds (implies password)

--principal *principal*

Use principal name

-d, --force

If an existing destination file cannot be opened remove it

-i, --interactive

Prompt before overwrite

-s, --status

Print information about each transfer.

-n, --no-stage

Read the file directly from tape if not staged. This flag has no effect if the source file is outside of HPSS.

-t, --no-tape

Do not copy the HPSS source file if it requires staging. This flag has no effect if the source is outside of HPSS.

-l, --to-local

Destination is local, source is HPSS.

-p, --to-hpss

Destination is HPSS, source is local.

-c, --cos *COSID*

Create the file in COS *COSID*

-f, --family *FAMILYID*

FAMILYID Create the file in file family *FAMILYID*

-u, --purge-on-migrate

Flag the files as needing to be purged immediately upon migrate

-x, --remove-destination

Remove each existing destination file before attempting to open it (contrast with force)

F.28.5. Example

Example 1: Copy a file from HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsscp --to-local /tmp/path/to/hpss/file /tmp/local/path
```

Example 2: Copy a file from HPSS using short flags

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsscp -l /tmp/path/to/hpss/file /tmp/local/path
```

Example 3: Copy a file from HPSS using short flags

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsscp -l /tmp/path/to/hpss/file /tmp/local/path
```

Example 4: Copy a file from HPSS using a stripe width of 2 and a block size of 5MB.

```
# export HPSS_API_PIO_STRIPE_WIDTH=2
# export HPSS_API_PIO_BLOCK_SIZE=5242880
# hpsscp -l /tmp/path/to/hpss/file /tmp/local/path
```

F.29. hpss_create_default_vv_select_weights(7)

F.29.1. Name

hpss_create_default_vv_select_weights - Create initial new volume selection weights based on the current configuration

F.29.2. Synopsis

hpss_create_default_vv_select_weights [-h]
[--output-file *OUTPUT_FILE*] [--verbose] [--weight-drive-types]
[--same-pvr-penalty *SAME_PVR_PENALTY*] --sc *SC*
[--add-drives *ADD_DRIVES*]

F.29.3. Description

This tool will create a PVR and drive default new volume selection weighing scheme based upon the HPSS system configuration. Volume selection here means the selection of a new tape volume for writing to a tape storage class - this tool does not alter or change the priority of volume selection for recalls or other operations. The weighting scheme is meant to be used as an input to HPSS tools that control new volume selection, such as hpss_update_priority_by_pvr or hpss_update_priority_by_pvr_drive.

This default scheme is meant to promote even use across PVRs, weighted by available drives within each PVR, rather than for example simply round robining between PVRs. A site may have a good reason to have a different scheme such as a true round robin or an adjusted weighing based on differential workloads between tape libraries or other factors. Sites which desire a different scheme may want to use this tool to quickly generate a template to begin modifying for their local needs.

The tool accepts a storage class as an input. It will identify the tape drive type directly associated with that storage class (e.g. E08 media would map to E08 drives). The tool will then scan the HPSS system configuration, including down PVRs or locked tape drives, looking for PVRs which contain drives of that type and counting them up. It will then spit out a configuration file which captures the weighting based on the total drive count.

Most tape media can only be written to by the current generation tape drive, however some can be written to by multiple generations. This tool does not currently attempt to make decisions based on that capability. However, the tool does allow for an admin to provide a set of additional drives to consider, and those drives will be included in the weighing scheme. These are provided by a comma separated list of drive IDs. So, if a site is planning to use legacy hardware to write current generation tape media, that can be provided as an input to the tool, but it will not be considered by default.

There are two basic modes of operation, the default will only provide weights by PVR based on the total drive count while an alternative mode weights by PVR and drive type. This can be used to cause HPSS to promote selecting more cartridges which are readable by one drive type over another (e.g. promote using remaining LTO-8 media over LTO-9 media).

Another option to note is the "same PVR penalty". This value represents the penalty that HPSS should apply when prioritizing striped tape volumes based on two or more tapes residing in the same tape PVR. This is meant to prioritize striped tape volumes which are in different tape libraries where possible because in that case the tape media can be moved to/from the tape drives in parallel, whereas that may not be true of tapes in the same library.

F.29.4. Required Arguments

--sc

Storage class to operate upon. The tool can only operate on one storage class (SC) at a time.

F.29.5. Optional Arguments

-h

Print the help output.

--verbose

Show additional information about steps being taken in generating the output.

--output-file OUTPUT_FILE

By default the tool's JSON output will go to stdout. If this argument is specified, it will go to the file identified as OUTPUT_FILE.

--weight-drive-types

Described in the description above, this changes the mode of operation from one where we generate weights based on PVR alone to one based on the PVR/DRIVE combination.

--same-pvr-penalty SAME_PVR_PENALTY

Described in the description above, this changes "Same PVR Penalty" which impacts the total weight given to an HPSS virtual volume which has two or more tapes in the same PVR.

--add-drives ADD_DRIVES

Described in the description above, this is a comma-separated list of HPSS Drive IDs which should also be included in the weighing. This can be used to include drive types which do not exactly match the storage class, if desired.

F.29.6. Prerequisites

The following software prerequisites are required:

- Python 3.6 or later
- the ibm-db Python package (available via *pip install ibm_db* or here: https://pypi.org/project/ibm-db/)

F.30. hpss_db_metrics(7)

F.30.1. Name

hpss_db_metrics - query the HPSS system for metrics from the HPSS database.

F.30.2. Syntax

hpss_db_metrics [-u *db_user*]
[-p *db_password*]
[-s *subsys_id*]
[-t *mm keytab path*]
[-j | -r]
[-f]
[-a] or [-l, -c, -m, -p]
[-h] [-?]

F.30.3. Description

Queries the database for various information (see below parameters) and can optionally output that data in JSON format.

F.30.4. Parameters

-s *subsys_id*, --subsys *subsys_id*

Subsystem ID of our database to query. If not specified, the default is 1.

-u *db_user*, --user *db_user*

User to query the database with. This user's db2profile is sourced to run queries. If not specified, it will be retrieved from the HPSS environment variable "HPSS_USER_DB".

-t *mmkeytab*

Used to specify the full path to the mm.keytab file. Defaults to the default value for the hpss_mm_keytab tool.

-j, --json

Output results in single JSON string

-r, --readable

Output results in human-readable JSON format

-f, --family

Used to output the results of files to migrate and/or purge by file family as well as the storage class. Not implemented for [-l and -c].

-a, --all

Retrieves database metrics of all commands. Do not use with [-l, -c, -m, -p] arguments

-l

Retrieves the count and total size of all purge-locked data, grouped by user and storage class

-c

Retrieves the count and total size of all trashcan data, grouped by user and COS

-m

Retrieves the count and total size of all files to migrate, grouped by storage class. When paired with -f will group by file family.

-p

Retrieves the count and total size of all files to purge, group by storage class. When paired with -f will group by file family.

-h, --help, -?

Display usage information.

F.30.5. Examples

To run all db metric queries using the HPSS environment to get the *db_user* and *db_password*

```
hpss_db_metrics -a
        [0]
        [0]
        [{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
        [{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
```

To run all db metric queries and output as single string JSON.

```
        hpss_db_metrics -a -j
        [[0], [0], [{"SCLASS_ID": 1, "TOTAL_BYTES": 4856248055, "FILE_COUNT": 5}],
[{"SCLASS_ID": 1, "TOTAL_BYTES": 4856248055, "FILE_COUNT": 5}]]
```

To run all db metric queries and output as human-readable JSON.

```
        hpss_db_metrics -a -r

        [
          [0],
          [0],
          [
            {
              "FILE_COUNT": 5,
              "SCLASS_ID": 1,
              "TOTAL_BYTES": 4856248055
            }
          ],
```

```
          [
            {
              "FILE_COUNT": 5,
              "SCLASS_ID": 1,
              "TOTAL_BYTES": 4856248055
            }
          ]
        ]
```

To run all db metric queries and output files to merge and purge by COS and file family.

```
hpss_db_metrics -a -f
        [0]
        [0]
        [{'SCLASS_ID': 1, 'FAMILY_ID': 0, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5},
        {'SCLASS_ID': 1, 'FAMILY_ID': 2, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
        [{'SCLASS_ID': 1, 'FAMILY_ID': 0, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
```

To run all db metric queries using the HPSS environment to get the *db_user* but passing in the *db_password*

```
        hpss_db_metrics -p <db_password> -a
        [0]
        [0]
        [{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
        [{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
```

To run all db metric queries passing in the *db_name* and *db_password*

```
        hpss_db_metrics -u <db_user> -p <db_password> -a
        [0]
        [0]
        [{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
        [{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
```

To retrieve count and size total of all trashcan data and purge-locked data.

```
       hpss_db_metrics -lt
       [0]
       [0]
```

To retrieve count and size total of all files to migrate, sorted by COS

```
      hpss_db_metrics -m
```

```
[{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
```

To retrieve count and size total of all files to purge, sorted by COS

```
hpss_db_metrics -p
[{'SCLASS_ID': 1, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5}]
```

To retrieve count and size total of all files to purge, sorted by COS and file family

```
hpss_db_metrics -p -f
[{'SCLASS_ID': 1, 'FAMILY_ID': 0, 'TOTAL_BYTES': 4856248055, 'FILE_COUNT': 5},
 {'SCLASS_ID': 1, 'FAMILY_ID': 1, 'TOTAL_BYTES': 1049600, 'FILE_COUNT': 2},
 {'SCLASS_ID': 1, 'FAMILY_ID': 2, 'TOTAL_BYTES': 83886080, 'FILE_COUNT': 1}
]
```

F.31. hpss_db_diag(7)

F.31.1. Name

hpss_db_diag - Query Db2 monitor routines and diagnostics

F.31.2. Syntax

hpss_db_diag [-d *db_name*]
[-u *db_user*]
[-t *mm keytab path*]
[-j | -r]
[-m *mntables*]
[-h] [-?]

F.31.3. Description

Queries the database for Db2 monitor routines to retrieve performance metrics such as active database connection, buffer pool activity, and locking information for diagnostic and tuning purposes.

F.31.4. Parameters

-d *db_name*, --database *db_name*

The database to query. If not specified the default is *hsubsys1*.

-u *db_user*, --user *db_user*

User to query the database with. If not specified, the user will be retrieved from the key tab specified by the HPSS environment variable "HPSS_SERVER_DB_KEYTAB"

-t *mmkeytab*

Used to specify the full path to the mm.keytab file. Defaults to the default value for the **hpss_mm_keytab** tool.

-j, --json

Output results in single JSON string

-m *mntable*, --mntables *mntable*

Used to specify which specific monitoring routine should be queried. Defaults to *ALL*, with *CONNECTION*, *DATABASE*, and *TABLE* as options.

-r, --readable

Output results in human-readable JSON format

-h, --help, -?

Display usage information.

F.31.5. Examples

If no options are provided, get all metrics from the default database. The user is retrieved from the HPSS environment.

```
% hpss_db_diag -m ALL
```

To get all metrics while specifying the db_name and db_password

```
% hpss_db_diag -m ALL <db_user> -p <db_password>
```

To get all metrics for a specific database

```
% hpss_db_diag -m ALL -d hcfg
```

To get information only about the client connection

```
% hpss_db_diag -m CONNECTION
APP_NAME         APP_HANDLE APP_ID                        CONN_START_TIME
ROWS_RETURNED DEADLOCKS ACT_ABORTED_TOTAL ACT_COMPLETED_TOTAL ACT_REJECTED_TOTAL
--------------- ---------- ----------------------------
----------------------------- ------------ -------- -----------------
------------------ ------------------
python3          3544       *LOCAL.hpssdb.250418154901     2025-04-18T10:49:01.434022
0             0         0                 0                   0
```

To get information only about the database

```
% hpss_db_diag -m DATABASE
DB_CONN_TIME         DB_STATUS DB_ACTIVATION_STATE  TOTAL_CONS CONNECTIONS_TOP
LAST_BACKUP          APPLS_CUR_CONS  NUM_LOCKS_HELD  NUM_LOCKS_WAITING
ACT_ABORTED_TOTAL ACT_COMPLETED_TOTAL ACT_REJECTED_TOTAL
------------------- --------- ------------------- ---------- --------------
------------------- -------------- -------------- --------------------
----------------- ------------------ ------------------
2025-04-17T10:21:19  ACTIVE    EXPLICIT             342        24              None
1               1               0                   0                209
0
```

To get information about all tables in the database

```
% hpss_db_diag -m TABLE
MEMBER TABSCHEMA  TABNAME                   TABSCANS   DBPARTITIONNUM TOTAL_ROWS_READ
TOTAL_ROWS_INSERTED  TOTAL_ROWS_UPDATED   TOTAL_ROWS_DELETED
------ ---------- ------------------------ ---------- --------------
------------------- ------------------- ------------------- --------------------
0      HPSS       ACCTLOGBANDWIDTH          0          0              0
```

```
0                    0                    0
```

Print the CONNECTION output in JSON only

```
% hpss_db_diag -m CONNECTION -j
{"MON_GET_CONNECTION": [{"APPLICATION_NAME": "python3", "APPLICATION_HANDLE": 3576,
"APPLICATION_ID": "*LOCAL.hpssdb.250418155622", "CONNECTION_START_TIME": "2025-04-18
10:56:18.808386", "ROWS_RETURNED": 0, "DEADLOCKS": 0, "ACT_ABORTED_TOTAL": 0,
"ACT_COMPLETED_TOTAL": 0, "ACT_REJECTED_TOTAL": 0}]}
```

Print the CONNECTION output in human-readable JSON

```
% hpss_db_diag -m CONNECTION -r
{
  "MON_GET_CONNECTION": [
    {
      "APPLICATION_NAME": "python3",
      "APPLICATION_HANDLE": 3567,
      "APPLICATION_ID": "*LOCAL.hpssdb.250418155547",
      "CONNECTION_START_TIME": "2025-04-18 10:55:47.279888",
      "ROWS_RETURNED": 0,
      "DEADLOCKS": 0,
      "ACT_ABORTED_TOTAL": 0,
      "ACT_COMPLETED_TOTAL": 0,
      "ACT_REJECTED_TOTAL": 0
    }
  ]
}
```

F.32. hpss_delog(7)

F.32.1. Name

hpss_delog - Retrieve and format HPSS log file entries

F.32.2. Syntax

hpss_delog [options] Logfile Delogfile
-s *StartTime*
-e *EndTime*
-d *Descriptive Name*
-t *Record Type*
-u *User*
-p *Principal*
-k *Keytab File*
-v *Log Version Number*

F.32.3. Description

"hpss_delog" is a legacy utility used to retrieve and format log records from old (pre-7.4) HPSS log files written in binary format. The program retrieves records from the binary log file (Logfile), converts them to a readable text format, and outputs the text to a specified file (Delogfile).

The binary central log files available for delogging were named logfile01 and logfile02. These files resided in the directory name specified in the Log Daemon configuration record (/var/hpss/log by default). Normally today, such files will be found in the HPSS archive where they were stored after being filled. Archived log files reside in a date-based HPSS directory structure: /log/%Y/%m/%d. To determine the log times for an archived log file, list the log files. When a log file is archived to HPSS, the current timestamp is appended to the log file name. The format of the archive file names is logfile01_%Y%m%d%H%M or logfile02_%Y%m%d%H%M. (See date(1) for an explanation of the notation %Y, %m, %d, etc.)

If no options other than the input log file pathname and output delog pathname are specified, all records in the log are delogged. Additional options allow filtering by start time (-s), end time (-e), server descriptive name(s) (-d), log record type(s) (-t), and user name(s) (-u). Command line arguments are described below. Options (arguments beginning with a hyphen) may precede or follow the positional arguments (Logfile, Delogfile).

The options -s (start time) and -e (end time) each take a time specification argument to determine which log records to delog. The following time formats are supported:

```
[%Y-]%m-%d %H:%M[:%S]
[%Y-]%m-%dT%H:%M[:%S]
[%Y-]%m-%dt%H:%M[:%S]
[%Y:]%m:%d:%H:%M[:%S]
```

Year and seconds are both optional. If year is omitted, the current calendar year is taken as the

default. If the seconds value is omitted, it defaults to 0.

F.32.4. Arguments

Logfile

The pathname of the input log file name. The log file may be the primary or secondary logfile (logfile01 / logfile02), or the name of a file retrieved from the archive, which can be anything.

Delogfile

The pathname of the output delog file.

-s *Start Time*

The time of the earliest record to retrieve. At most, one -s option may be entered. The default is the earliest record in the log.

-e *End Time*

The time of the latest record to retrieve. At most, one -e option may be entered. The default is the latest record in the log.

-d *Descriptive Name*

A server descriptive name associated with the log records to retrieve. Multiple -d options may be specified. Partial matches are supported. For example, *-d Mover* would match *Mover 123*, and *Mover abc*, but not *Movie*. The default is all servers.

-t *Record Type*

A log record type associated with the log records to retrieve. Allowed values are alarm, event, request, security, accounting, debug, and trace. Multiple -t options may be specified. The default is all record types.

-u *User*

A user name associated with the log records to retrieve. Multiple -u options may be specified. The default is all users. It should be noted that user names are not generally available from the core HPSS servers. Partial matches are supported.

-p *Principal*

The principal associated with the delog process. If not specified, the value will be obtained from the HPSS_PRINCIPAL_LOG environment variable if defined or default to *hpss_log*. Note: If not specified, the principal will be obtained (in order) from the HPSS_LOG_PRINCIPAL environment variable, the HPSS_PRINCIPAL_LOG environment variable, or default to *hpss_log*. This option is supported for backward compatibility but is not used in the 7.4 version of the program. Originally it was used to connect to HPSS to delog files directly from the archive.

-k *Keytab File*

The keytab file associated with delog principal. If not specified, the value will be obtained from the HPSS_KEYTAB_FILE_SERVER environment variable if defined or default to */krb5/hpss.keytabs*. Note: If not specified, the keytab pathname will be obtained (in order) from the HPSS_LOG_KTAB_PATH environment variable, the HPSS_KEYTAB_FILE_SERVER environment variable, or default to */krb5/hpss.keytabs*. This option is supported for backward

compatibility but is not used in the 7.4 version of the program. Originally it was used in connecting to HPSS to delog files directly from the archive.

-v *Log Version*

The log version to use when delogging log records. If not specified, the value will be obtained from the header of the first log record processed. Allowed values are 1 and 2.

F.32.5. Examples

Delog all records from log file /var/hpss/log/logfile01 to output file /hpss/data/delog.out.

```
hpss_delog /var/hpss/log/logfile01 /hpss/data/delog.out
```

Delog records between 1:00pm and 3:00pm on March 1, 1999.

```
hpss_delog /var/hpss/log/logfile02 /hpss/data/delog.out \
           -s1999:03:01:13:00:00 -e1999:03:01:15:00:00
```

Delog alarm and debug records for server PVL between 1:00pm and 3:00pm on March 1, 1999.

```
hpss_delog /var/hpss/log/logfile02 /hpss/data/delog.out \
           -s1999:03:01:13:00:00 -e1999:03:01:15:00:00 -talarm -tdebug -dPVL
```

Delog only alarm records.

```
hpss_delog /var/hpss/log/logfile01 /hpss/data/delog.out -talarm
```

Delog alarm and event records for the Core Server between 8:00am and 9:30am on February 28. The log file referenced has been archived to hpss.

```
hpss_delog /log/1999/02/28/logfile01_199902280745 \
           /hpss/data/delog.out -s02:28:08:00:00 -e02:28:09:30:00 \
           -talarm -tevent -d'Core Server'
```

Delog records using log version 2.

```
hpss_delog /var/hpss/log/logfile02 /hpss/data/delog/out -v2
```

F.32.6. Limitations

Past versions of hpss_delog were able to read binary log files directly from the HPSS archive. With HPSS release 7.4, the program was rewritten in Python, which does not support direct access to the HPSS archive at this time.

F.33. hpss_drive_ctl(7)

F.33.1. Name

hpss_drive_ctl - Query or alter the administrative state of an HPSS drive

F.33.2. Syntax

hpss_drive_ctl *enable*|*disable*|*query drive ID*
[-m]
[-w *TIMEOUT*]
[-p *principal*]
[-k *authenticator*]
[-g *HPSS global config db name*]
[-s *HPSS db schema name*]
[-m *HPSS Authentication Mechanism*]

F.33.3. Description

hpss_drive_ctl is a tool for querying or setting the Administrative state of an HPSS drive.

The three supported operations are "query" to just report the state of the drive, "enable" to unlock a locked drive, and "disable" to lock an unlocked drive. Note that for the enable operation, the program assumes that the drive is locked to begin with, and for the disable operation, the program assumes that the drive is unlocked to begin with.

F.33.4. Options

drive ID

This is the numeric HPSS drive ID of the drive to control.

-M

"Monitor" mode: the program will continuously print the drive's state once a second

-w *TIMEOUT*

Wait (for no more than TIMEOUT seconds) after locking the drive for the Drive state to become FREE. Note that if the drive does not become free within the specified timeout after being locked, that the program will then unlock the drive and exit. A timeout of -1 means to wait indefinitely. This option is only valid when the selected operation is "disable".

-p *principal*

The principal to use when communicating with the PVL. If not specified, the utility will use the value in the environment variable HPSS_PRINCIPAL_SSM.

-k *authenticator*

The authenticator to use to authenticate as the principal. If not specified, the utility will use the value in the environment variable HPSS_PRIMARY_AUTHENICATOR.

-g global config db name

The name of the HPSS global config database. If not specified, the utility will use the value in the environment variable HPSS_GLOBAL_DB_NAME.

F.33.5. Examples

To report on the state of drive ID 1:

```
$ hpss_drive_ctl query 1

<produces something like...>

        DriveID         1
        AdminState      unlocked
        DriveState      in use
```

To monitor the state of a drive continuously:

```
$ hpss_drive_ctl -m query 1

        DriveID         1
        AdminState      unlocked
        DriveState      free

        ...(prints these out every second)...
```

To lock a drive and wait for it to become free for 100 seconds:

```
$ hpss_drive_ctl disable 1 -w 100

        ...(as soon as the drive becomes free)...

        DriveID         1
        AdminState      locked
        DriveState      free
```

F.34. hpss_errno(7)

F.34.1. Name

hpss_errno - HPSS error code tool

F.34.2. Syntax

hpss_errno [OPTION] *ERRORCODE* ...

F.34.3. Description

The hpss_errno tool can be used to translate HPSS error code values into strings.

F.34.4. Options

-n, --name

Also print out the HPSS error code name. For instance, the error code name of 0 is "HPSS_E_NOERROR".

F.34.5. Examples

```
% /opt/hpss/bin/hpss_errno 1
1=Operation not permitted

% /opt/hpss/bin/hpss_errno -1604
1604=Resource not mounted

% hpss_errno 1000 1001 1002
1000=Group vector too long
1001=Path crosses junction
1002=Invalid configuration bit

% hpss_errno -n -5
-5=I/O error (HPSS_EIO)
```

F.35. hpssfs(7)

F.35.1. NAME

hpssfs - mount HPSS as a local filesystem

F.35.2. SYNOPSIS

hpssfs [options] *target mountpoint*

F.35.3. DESCRIPTION

hpssfs is a program to mount HPSS as a local filesystem.

F.35.4. OPTIONS

F.35.4.1. hpssfs-specific options

-h, --help
print help.

-V, --version
print version.

-o attrtimeo=*SECONDS*
Time to keep cached file attributes [60].

-o entrytimeo=*SECONDS*
Time to keep cached directory attributes [30].

-o stagetimeo=*SECONDS*
Time to wait for stage completion [3600].

-o udatimeo=*SECONDS*
Seconds to wait for UDA lock (minimum allowed value 25) [25].

-o cos=*COSID*
COS ID on newly created files [HPSS-determined].

-o family=*FAMILYID*
Family ID on newly created files [none].

-o trace=*LEVEL*
Level of detail of logging [0].

-o auth=*AUTHENTICATOR*
Primary authenticator [HPSS_PRIMARY_AUTHENTICATOR].

-o authmech=***MECHANISM***

Primary authentication mechanism [HPSS_PRIMARY_AUTHN_MECH].

-o authtype=***TYPE***

Authentication type [derived from *auth* value].

-o princ=***USER***

Principal name [HPSS_PRINCIPAL_FS].

-o ip=[*HOSTNAME*|*IP*|*DEVICE*]

Add a NIC for striped I/O [HPSS_API_HOSTNAME]. May be supplied up to 32 times.

-o ctrlpath=[*HOSTNAME*|*IP*|*DEVICE*]

Specify interface for Core server staging operations.

-o stream=***MB***

Buffersize for readahead/writeback [8].

-o nostream

Use unbuffered I/O (equivalent to -ostream=0).

-o maxfsz=***MB***

Maximum offset to allow writing (0 means unlimited) [0].

-o autopurgelock=***BYTES***

Maximum file size to auto-purge-lock (0 means disabled) [0].

-o [no]purgeonmigr

Files created with purge on migrate flag [nopurgeonmigr].

-o idmap=***MODE***

Enable ID mapping [none]. MODE can be: **none**, **user**, **file**.

-o uidfile=***PATH***

UID mapping file; only used with idmap=file.

-o gidfile=***PATH***

GID mapping file; only used with idmap=file.

-o [no]acl

Enable POSIX ACL extended attributes [noacl].

-o [no]san

Enable SAN3P support [HPSS_API_SAN3P].

-o [no]shm

Enable shared memory transfers [noshm].

-o [no]stage

Enable stage on open [stage].

-o [no]stagetape

Enable mount point to open files on tape [stagetape].

-o eremote_callout=*FILE*

Callout/hook for EREMOTE with nostagetape.

-o [no]dio

Enable Direct I/O [nodio].

-o [no]maxsegsz

Set maxsegsz COS hint on file creation [nomaxsegsz].

-o [no]nfs4

Enable NFSv4 export optimizations [nonfs4].

-o [no]encrypt

Enable encryption for data transfers [HPSS_API_ENCRYPT]

-o cksum=*ALGORITHM*

Enable checksum processing [none]. ALGORITHM can be: **none**, **crc32**, **adler32**, **md5**, **sha1**, **sha224**, **sha256**, **sha384**, **sha512**.

-o nch=*ACTION*

Action when checksum enabled and file lacks checksum attributes [f]. ACTION can be: **f** (Fail), **g** (Generate), **i** (Ignore).

-o ckstyle=*STYLE*

Where to store checksum attributes [hybrid]. STYLE can be: **filehash**, **uda**, **hybrid**.

-o rvl=*SECONDS*

Require checksum revalidation after SECONDS [0].

-o [no]cksumatime

Allow checksum readbacks to update atime [cksumatime].

-o [no]stagetape

Allow HPSSFS to stage files from tape on access

-o eremote_callout=FILE

Callout/hook for when an EREMOTE occurs with nostagetape. FILE is a valid executable.

-o [no]stickytape

Report the sticky bit set for files on tape

target

HPSS target to mount. May be HPSS path or Fileset ID in format *high,,low* (e.g., `3272743334,,4016631840` or `3272743334,,4016631840/subpath`). Either may be prefixed with `sitename:` (default is LOCAL:).

mountpoint

Local directory to mount.

F.35.4.2. Supported FUSE mount options

See **fuse**(8) for details.

-d

Enable FUSE debugging.

-f

Run in foreground.

-s

Make FUSE requests single-threaded.

-o allow_other

Allow other users to access the mount point.

-o allow_root

Allow root user to access the mount point.

-o max_background=#

Controls number of background threads [default 100].

-o congestion_threshold=#

Controls threshold for congestion [default 75% of max_background].

-o auto_unmount

Automatically unmount if process dies.

-o debug

Enable FUSE debugging (same as -d).

-o nonempty

Allow mount even if mount point is not empty.

-o readdirplus=[auto|no|yes]

Enable readdirplus.

F.35.4.3. Supported kernel mount options

See **mount**(8) for details.

-o ro

Mount as read-only.

-o rw

Mount as read-write.

-o [no]atime

Update file access times.

-o [no]dev

Allow access to special devices.

-o [no]exec

Allow programs to be executed.

-o [no]suid

Honor suid-bit on programs.

-o [a]sync

Perform I/O asynchronously or synchronously.

-o dirsync

Complete all directory updates synchronously.

-o context=*context*, defcontext=*context*, fscontext=*context*, rootcontext=*context*

Set SELinux contexts.

F.35.5. EXAMPLES

Specify multiple options

```
hpssfs -ocos=1,stream=16 / /mnt/hpss
```

Specify a target which is not the root directory

```
hpssfs /home/me /home/me/hpss
```

Use Fileset ID as target

```
hpssfs 3272743334,,4016631840 /mnt/hpss
```

Use a non-default site name

```
hpssfs MySite:/ /mnt/hpss
```

/etc/fstab entry

```
/ /mnt/hpss fuse.hpssfs cos=1,stream=16 0 0
```

F.35.6. SEE ALSO

fuse(8), **fusermount**(1), **fstab**(5), **hpssfs_undelete**(7), **mount**(8)

F.36. hpssfs_undelete(7)

F.36.1. NAME

hpssfs_undelete - Undelete files and directories

F.36.2. SYNOPSIS

hpssfs_undelete [OPTION]... *FILE*...

F.36.3. DESCRIPTION

hpssfs_undelete is a program to undelete files and directories in HPSS. See HPSS documentation for more information about undelete.

F.36.4. OPTIONS

-h, --help
print help.

-o, --overwrite
overwrite an existing file or directory.

-r, --restore-time
restore timestamps.

-v, --verbose
print files and directories which have been undeleted.

F.36.5. EXAMPLES

Undelete a file or directory

```
hpssfs_undelete /mnt/hpss/.Trash/root/file1.999
```

Undelete a file or directory, with overwrite and restore time

```
hpssfs_undelete --overwrite --restore-time /mnt/hpss/.Trash/root/file1.999
```

Undelete multiple files or directories

```
hpssfs_undelete /mnt/hpss/.Trash/root/file1.999 /mnt/hpss/.Trash/root/file2.1000
```

F.36.6. NOTES

This functionality requires the HPSS Trashcan feature. The ability to undelete directories additionally requires libfuse >= 2.9 and Linux kernel >= 3.3.

F.36.7. SEE ALSO

hpssfs(7)

F.37. hpss_ftp(7)

F.37.1. Name

hpss_ftp - HPSS Parallel FTP (PFTP) Client and Daemon

F.37.2. Description

Non-HPSS FTP (non-Parallel) clients will, with a few exceptions, function with the HPSS Parallel FTP Daemon with degraded performance. The HPSS project cannot assure complete compliance. Non-standard implementations may encounter errors and support from the HPSS Project is NOT assured.

Refer to appropriate documentation for the particulars on non-HPSS ftp clients.

The HPSS PFTP Client provides the required functions of the standard FTP Client plus additional features providing parallel network functions when communicating with the HPSS Parallel FTP Daemon.

The HPSS PFTP Daemon provides the mandatory functions of the standard FTP Daemon plus additional features providing parallel network functions when communicating with the HPSS PFTP Client.

The end user's home directory is read from either the authorization database (password style file) or the LDAP Registry. It is necessary to create the appropriate home directories in HPSS for each customer.

The pftp_client code is provided for AIX, Linux, and Solaris. Successful ports have been built for: SGI (32 and 64 Bit Mode), HP HPux 11, Cray Unicos, and DEC Alpha Digital Unix, but are no longer supported. Customer sites may port to other platforms, the HPSS Project cannot assure support.

Refer to the HPSS Users Guide and the man pages (SEE ALSO) for detailed descriptions of PFTP features.

The current implementations of the HPSS FTP Daemon supports authentication of customers using two methods:

1. Standard Username/Password authentication using a password file
2. Using the Generic Security Service (GSS) based FTP given Kerberos (Kerberos kinit).

F.37.3. Building a Configuration File

A configuration file will be used to specify different IP addresses or remote nodes on which the data transfer processes are to be spawned by HPSS Parallel FTP Client (multinode mode) or which network interfaces to use for network striping (multihomed mode). The default configuration file is /var/hpss/etc/HPSS.conf. The user can override the path to this file with the HPSS_CFG_FILE_PATH environment variable. It is the Administrator's responsibility to correctly configure and maintain the HPSS.conf file as well as implementing the multinoded components on all remote systems.

F.37.4. Files

/etc/inetd.conf

/etc/services

/var/hpss/etc/HPSS.conf

/var/hpss/ftp/adm/hpss_ftpd.log

/var/hpss/ftp/adm/xferlog

/var/hpss/etc/ftpaccess

/var/hpss/etc/ftpbanner

/var/hpss/etc/ftpusers

/var/hpss/etc/passwd

/var/hpss/etc/shadow

/var/hpss/etc/group

/var/hpss/etc/authz.conf

~/.netrc

Appropriate Shared Libraries

F.37.5. See Also

ftp,HPSS.conf,hpss_pftpd,multinoded,netrc,pftp_client,syslog,xinetd

HPSS User's Guide

HPSS Admin Guide

F.37.6. Author

Tim Preston, Sandia National Labs, Albuquerque, NM

F.38. hpssgetvol(7)

F.38.1. Name

hpssgetvol - Get volume information for an HPSS file

F.38.2. Syntax

hpssgetvol *FILE...*

F.38.3. Description

This program lists the storage level information of one or more HPSS files.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.38.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.38.5. Example

Example 1: Print volume information for an HPSS file

```
# export HPSS_CLI_USER=myuser
```

```
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssgetvol /path/to/hpss/file
Level 0: Bytes = 1024
Level 0: VV[0]: Relative Position = 1
Level 0: PV[0]: ABC12300
```

F.39. hpssgetxattrs(7)

F.39.1. Name

hpssgetxattrs - Get extended attributes for one or more HPSS files

F.39.2. Syntax

hpssgetxattrs [*OPTION*] *FILE*...

F.39.3. Description

This program prints extended attributes for HPSS files. The output can include level information (--level), first level information (--flevel), optimization (--opt), or all (--all). The input must contain the storage level to retrieve information for (0 being the top level) and a file name.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

See the HPSS Admin Guide for a description of fields in this output.

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

-a, --all
Print all statistics.

-l, --level

Print level statistics.

-f, --flevel

Print first level statistics.

-o, --opt

Prints only StripeWidth and OptimumAccessSize for storage level zero.

F.39.4. Example

Example 1: Get extended attributes on a file

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssgetxattrs /test1
File Name                      : /test1
Account                        : 0
RealmId                        : 10000
Comment                        : NONE
CompositePerms                 : 355
CosId                          : 2
BitfileId: ServerId            : ServerId 2
BitfileId:                     : 00000315-07-00000002-01ecd78abce0aad0-0000
Bitfile Hash:                  : 14968
Bitfile Hash Valid:            : TRUE
DataLength                     : 1 MB
DontPurge                      : Off
SuperDontPurge                 : Off
PurgeOnMigrate                 : Off
EntryCount                     : 0
ExtendedACLs                   : 0
FamilyId                       : 0
FilesetHandle ObjId            : 0
FilesetHandle FileId           : 0
FilesetHandle Type             : 0
FilesetHandle Flags            : 0
FilesetHandle Generation       : 0
FilesetHandle CoreServerId     : Server: 0
FilesetId                      : 3273054433.3142149520
FilesetRootObjectId            : 0.1
FilesetStateFlags              : 3
     CORE_FS_STATE_READ        : ON
     CORE_FS_STATE_WRITE       : ON
     CORE_FS_STATE_DESTROYED   : OFF
FilesetType                    : HPSS Only
GID                            : 0
GroupPerms                     : 140
LinkCount                      : 1
OpenCount                      : 0
```

```
OtherPerms                    : 40
ReadCount                     : 314436
RegisterBitMap                : 00000000.00000000
SetUIDBit                     : Off
SetGIDBit                     : Off
SetStickyBit                  : Off
TimeCreated                   : Thu May 19 10:45:41 2022
TimeLastRead                  : Tue Jun 28 13:47:59 2022
TimeLastWritten               : Thu May 19 10:45:43 2022
TimeModified                  : Thu May 19 10:45:43 2022
Trash ParentId                : 0
Trash ParentNsHash            : 0
Trash Handle ObjId            : 0
Trash Handle ObjNsHash        : 0
Trash Handle FileId           : 0
Trash Handle FileNsHash       : 0
Trash Handle Type             : 0
Trash Handle Flags            : 0
Trash Handle Generation       : 0
Trash Handle CoreServerId     : 0
Trash UID                     : 0
Trash Time Deleted            : Wed Dec 31 18:00:00 1969
Trash Time Created            : Wed Dec 31 18:00:00 1969
Trash Time Last Read          : Wed Dec 31 18:00:00 1969
Trash Time Last Modified      : Wed Dec 31 18:00:00 1969
Trash Path                    :
Trash File Name               :
Type                          : File
UID                           : 0
UserPerms                     : 141
ModePerms                     : 0
WriteCount                    : 1
SubSystemId                   : 1
ActiveWrites                  : -1
ActiveReads                   : -1
ActiveMigrates                : -1
ActiveModify                  : -1
ActiveCopy                    : -1
ActiveStages                  : -1
```

Example 2: Get level statistics on a file (no auth options)

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssgetxattrs -l /test1
```

Example 3: Get level statistics on a file (password auth)

```
# export HPSS_CLI_USER=myuser
```

```
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssgetxattrs --newcred -l /test1
```

F.40. hpssgui(7)

F.40.1. Name

hpssgui - The generic name (without programming language extension) for the SSM graphical user interface startup script.

F.40.2. Syntax

hpssgui.[pl|vbs]
[-A *"Alarm/Event refresh rate"*]
[-C *"login.conf file"*]
[-D *"Date format pattern"*]
[-F *"look and feel"*]
[-G *"Alarm/Event get number"*]
[-L *"List refresh rate"*]
[-M *"MO refresh rate"*]
[-N *"Alarm/Event display size"*]
[-P *"hpss.jar file"*]
[-S *"Session log"*]
[-T *"Theme file"*]
[-W *"Wait time"*]
[-a *"Authenticator"*]
[-b *"Background color"*]
[-c *"Country code"*]
[-d]
[-f *"Alternate help file path"*]
[-g *"Help path URL type"*]
[-h *"System Manager host name"*]
[-i *"SSM preferences path"*]
[-j *"Java binary path"*]
[-k *"krb5.conf file"*]
[-l *"Language code"*]
[-m *"SSM configuration file"*]
[-n *"port number"*|*"program number"*:*"program version number"*]
[-p *"Client IP address"*]
[-r *"RPC protection level"*]
[-s *"Security mechanism"*]
[-u *"UNIX Realm"*]
[-x]
[--property *"Java property string"*]

hpssgui.pl

Startup script for the SSM graphical user interface written in Perl.

hpssgui.vbs

Startup script for the SSM graphical user interface written in Microsoft Visual Basic Scripting

language.

Parameters

Note that the parameters for the hpssgui utility are the same as those for the hpssadm utility with the following exceptions:

The -F, -M, -T, -f, and -g options are supported only by the hpssgui utility, not by hpssadm.

The -U option (not listed above) is supported only by the hpssadm utility, not by hpssgui.

The -b option is supported by both the hpssgui and hpssadm utilities but is interpreted differently by each.

For each of the parameters, the associated environment variable name and configuration file variable name (see option -m below) are listed in the table in the ENVIRONMENT section below. More details about the architecture of the environment variables and configuration file variables are described in the following sections.

In the parameter descriptions below, notations like "$HPSS_USER_CFG_PATH" refer to environment variables, not Perl variables in the script.

Environment variables and pathnames are given in Unix format. When run under Windows, appropriate adjustments will be needed in environment variables (eg., %HPSS_USER_CFG_PATH% rather than $HPSS_USER_CFG_PATH).

-A *Alarm/Event refresh rate*

The number of seconds to wait between refreshes of the Alarms and Events list.

The default is 5 seconds.

See more information about performance considerations in the NOTES section below.

-C *login.conf file*

Specifies an alternate HPSS login configuration file.

The default is included in the hpss.jar file and is created from the template file in /opt/hpss/config/templates/login.conf.template.

Please see the /opt/hpss/config/templates/login.conf.template file for a description of the format and options.

Note that $HPSS_PATH_SSM is also used to locate the ssm configuration file and that it only specifies the directory. If the environment variable is used to locate the file, the file name itself is fixed as "login.conf". To use a different file name, it must be specified on the command line, or set in the configuration file.

-D *Date format pattern*

Specifies the format pattern for displaying dates.

Example: "kk:mm:ss dd-MMM-yyyy"

This will produce dates formatted as:

13:22:59 03-Jan-2004

For a complete list of the valid formatting options see the Java web page:

http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html

The default date format is based on the default Java date/time formatter for the user's locale.

-F *Look and feel*

The look and feel setting for the application. Java has several built-in look and feels for various platforms: Windows, Mac, Motif, Metal and GTK.

The valid command line arguments for the look and feel are: windows, mac, motif, metal and gtk

Because of copyright issues the Windows look and feel will only work on PCs running the Windows OS. Likewise, the Mac look and feel will only work on Macintosh computers.

Also note that the gtk look and feel does not work for the runtime version of java (jre) on Windows platforms. Windows users specifying the gtk look and feel can use the software development kit (sdk) version of java or install the look and feel separately.

The default look and feel is determined by the native look and feel for the platform that the gui is running on. For UNIX, the default is metal.

There are also lots of custom Java look and feels available.

-G *Alarm/Event get number* The maximum number of Alarms and Events to grab with each refresh of the Alarms and Events list.

The default is 500.

-L *List refresh rate* The number of seconds to wait between refreshes of lists. (Applies to all SSM window lists except the Alarms and Events list.)

The default is 5 seconds.

See more information about performance considerations in the NOTES section below.

-M *MO refresh rate*

The number of seconds to wait between refreshes of Managed Objects. (Includes the Health and Status Window.)

Managed Objects are windows for which the System Manager registers for updates to fields in a window with the associated server(s). For example, the Core Server Information window is a Managed Object; the Operational State may change and when it does, the System Manager will get this update from the Core Server.

The default is 5 seconds.

See more information about performance considerations in the NOTES section below.

-N *Alarm/Event display number*

The maximum number of alarms/events to display in the Alarms and Events list.

The default is 2000.

-P *hpss.jar file*

Specifies the location of the hpss.jar file.

Defaults to hpss.jar file in the current directory. If that does not exist then the script will try using environment variable $HPSS_SSM_CLASSPATH as the full path of the file.

-S *Session log file*

Specifies a pathname to an hpssgui session log file. This is extremely useful for diagnosing problems. This option should be used when using the -d option.

By default, session logging is off.

-T *Metal theme file*

A file containing the theme settings for the Metal look and feel.

There are 8 color parameters that can be set to customize the look of the window. These consist of 3 primary, 3 secondary and the black and white colors. The settings for these colors can be stored in a theme file using the following syntax:

```
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.primary1=COLOR
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.primary2=COLOR
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.primary3=COLOR
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.secondary1=COLOR
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.secondary2=COLOR
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.secondary3=COLOR
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.black=COLOR
hpss.ssm.ssmuser.hpssgui.CustomMetalTheme.white=COLOR
```

where COLOR is specified either by name or hexadecimal RGB (red/green/blue) value as for -b option below.

If not specified on the command line, the default is $HOME/hpss-ssm-prefs/DefaultTheme

Currently the only look and feel that supports color setting and themes is the Metal look and feel.

For a description of each of the color properties and how they affect the window see:

http://java.sun.com/products/jlf/ed1/dg/higg.htm

-W *Wait time* The number of seconds to wait for a managed object or list update from the System Manager.

The default is 5 seconds.

See more information about performance considerations in the NOTES section below.

-a *Authenticator*

The keytab file containing the user's encrypted keytab. If not specified, the default is a mechanism-dependent default in the user's home directory.

Default: ~/keytab (UNIX) or ~/krb5.keytab (KRB5)

-b *"Background color"*

The background color of the windows.

Colors can be specified by name or hexadecimal RGB (red/green/blue) values. Each of the R, G and B components is specified in the range of 0 to 255 (0x00 to 0xff). Here are some examples:

```
Name        Hex         Name        Hex
-----       --------    -------     --------
red         0xff0000    cyan        0x00ffff
green       0x00ff00    yellow      0xffff00
blue        0x0000ff    magenta     0xff00ff
```

By varying the RGB components many colors can be generated.

When the -b option is specified this script will set the look and feel to the Metal look and feel.

Currently the only look and feel that supports color setting and themes is the Metal look and feel.

The default background color is that associated with the native look and feel for the client platform.

-c *Country code*

Specifies the National Language Support (NLS) country code to be used by the application when displaying message text.

The default country code is taken from the system properties defined by Java.

-d

Turn on debug preference. Specify the -S option to see the debug level logs in a session log file.

By default, debug is turned off.

-f *Alternate help file path*

Specifies the location of the HPSS help files.

The default help file path is "/opt/hpss/ssmhelp". Note that on Windows clients, this path should be defined on the command line, in the user environment, or in the user's configuration file since the default path uses UNIX file path punctuation.

-g *Help path URL type*

The Uniform Resource Locator (URL) type of the help path. Currently, the only valid type is "file:"

The default type is "file:".

-h *System Manager host name*

Specifies the host name of the node on which the System Manager is running.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.

-i *SSM preferences path*

Specifies the path to search for application preferences and configuration.

The default configuration path is the "hpss-ssm-prefs" directory under the user's home directory path as defined by the Java property user.home.

-j *Java binary path* Specifies the path to the java binary.

If not specified on the command line, in the user's environment variable $JAVA_BIN, or in the configuration file, the script will look for a java binary in the user's $PATH. If one does not exist, the script will fail.

-k *krb5.conf file* Specifies an alternate Kerberos configuration file which allows the client to authenticate to an alternate Kerberos realm.

Defaults to krb5.conf in the current directory. If that does not exist then it will try to use $KRB5_CONFIG.

If the security mechanism (command line option "-s") is set to "krb5", then this file is required.

-l *Language code* Specifies the NLS language code to be used by the application when displaying message text.

The default language code is taken from the system properties defined by Java.

-m *SSM configuration file* Specifies the configuration file that defines the required variables for the SSM client to run properly. Users can also specify other variables in this file as opposed to specifying them on the command line.

The default file used is ssm.conf in the current directory. If that does not exist then the script will try to use $HPSS_PATH_SSM/ssm.conf.

Note that $HPSS_PATH_SSM is also used to locate LOGIN_CONF and that it only specifies the directory. If the environment variable is used to locate the file, the file name itself is fixed as "ssm.conf". To use a different file name, it must be specified on the command line, or set in the configuration file and HPSS_PATH_SSM left unset (although it doesn't make much sense to set it in the configuration file since this setting's job is to specify the location of the configuration file).

-n *port number* or *program number":"program version number*

Specifies the port number that the System Manager is using to accept incoming connections.

OR:

Specifies the ONC RPC program number and version that the System Manager is using to register

the port to be used to accept incoming connections.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.



The HPSS_SSM_SERVER_LISTEN_PORT can be set to a specific port upon which the site wants the System Manager to listen for client RPCs. If this is 0, then the port will be chosen by portmapper. Setting this to a non-zero value will give client scripts that need to go through a firewall a static port number that the System Manager is listening on; otherwise, the port will change each time the System Manager is cycled.

-p *Client IP address*

Specifies the IP address of the client machine. This should be used when the client is assigned a NAT address from a VPN server for example.

The default address is taken from the local host setting.

-r *RPC protection level*

Specifies the RPC protection level to use when connecting to the System Manager.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.

-s *Security mechanism*

Specifies the security mechanism to use when connecting to the System Manager.

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line. At install time, mkhpss will insert the appropriate value in the SSM configuration file.

-u *UNIX Realm*

Specifies the UNIX realm. This is often the lowercase fully qualified hostname of the SSM server.

This parameter is only needed if the security mechanism (command line option "-s") is set to "unix".

No default value is provided. This parameter must be defined in the SSM configuration file, in the user's environment, or on the command line if the security mechanism is "unix". At install time, mkhpss will insert the appropriate value in the SSM configuration file.

-x

Test mode. Just displays the Java command without actually executing it. This option is only supported in the .pl version (not .vbs).

The default value for test mode is off.

--property *Java property string*

Arbitrary property strings can be passed to the Java program being launched using the --property option. Property strings have the form "-D<property>=<value>". Multiple property strings separated by spaces can be passed to the --property option. For example:

```
... --property "-Dhpss.ssm.SMConnections=1 -Dother=different" ...
```

Properties that can be set in this way include

hpss.ssm.SMConnections=*number* The number of connections to open to the System Manager.

hpss.ssm.mobjects.RPCLoginContext.trace=*true*|*false*
hpss.ssm.mobjects.RPCInterface.trace=*true*|*false* hpss.jgss.GSSCredentialKRB5.trace=*true*|*false*
hpss.jgss.GSSMechanismKRB5.trace=*true*|*false* hpss.jgss.GSSMechanismManager.trace=*true*|*false*
hpss.jgss.GSSMechanismUnix.trace=*true*|*false*

Setting these properties to true sends trace output for the methods of each class (and its inner classes) to stdout. Setting them to false suppresses the output. Trace information can be very helpful for trouble-shooting, however bear in mind that collecting trace information can consume considerable disk space, depending upon the number of classes being traced and the load on the system.

hpss.ssm.mobjects.RPCLoginContext.debug=*true*|*false*
hpss.ssm.mobjects.RPCInterface.debug=*true*|*false* hpss.jgss.GSSCredentialKRB5.debug=*true*|*false*
hpss.jgss.GSSCredentialKRB5.debug=*true*|*false* hpss.jgss.GSSMechanismManager.debug=*true*|*false*
hpss.jgss.GSSMechanismUnix.debug=*true*|*false*

Setting these properties to true sends debug output for the methods of each class (and its inner classes) to stdout. Setting them to false suppresses the output. Debug information can be very helpful for trouble-shooting, however bear in mind that collecting debug information can consume considerable disk space, depending upon the number of classes being debugged and the load on the system.

com.ibm.security.jgss.debug=all When the client runs on an AIX platform, this property enables all categories of debug output for Java's GSS package. The following table shows the various categories that can be requested:

Category	Description
help	List debug categories
all	Turn on debugging for all categories
off	Turn off debugging completely
app	Application debugging (default)
ctx	Context operations debugging
cred	Credentials (including name) operations
marsh	Marshaling of tokens

mic	MIC operations
prov	Provider operations
qop	QOP operations
unmarsh	Unmarshalling operations
unwrap	Unwrap operations
wrap	Wrap operations

com.ibm.security.krb5.Krb5Debug=all

When the client runs on an AIX platform, this property enables debug output for Java's Kerberos package. The only supported values for this property are "all" and "off".

F.40.3. Description

The hpssgui scripts (hpssgui.pl and hpssgui.vbs) start the SSM Graphical User Interface. They bring up the SSM login window. When the user successfully types his/her name and password into this window, the program opens the other SSM windows. See the HPSS Admin Guide for details of using the windows and menus of the Graphical User Interface.

All command line options are optional if running in the default environment. Defaults are assigned based on SSM configuration file settings (ssm.conf) and environment variables. Here are some key default assumptions:

java binary location

If the java binary location isn't specified by the -j command line option, $JAVA_BIN user environment variable or SSM configuration file setting (JAVA_BIN), then it is assumed that it is in the client's $PATH.

SSM configuration file

The SSM client scripts will look for a file named "ssm.conf" in the current directory. If that does not exist then they will try to use $HPSS_PATH_SSM/ssm.conf (which is generated by mkhpss). The SSM client scripts will not run without a configuration file.

login configuration file

If the login configuration file isn't specified by the -C command line option or in the SSM configuration file (LOGIN_CONFIG), then the SSM client will use the login.conf file that is contained in the hpss.jar file. If the hpss.jar file does not contain a login.conf file, the SSM client will look for a file named "login.conf" in the current directory.

All hpssgui users must be listed as authorized users in the AUTHZACL table (see the "FILES" section of this man page).

The hpssgui.[pl|vbs] programs are scripts which invoke the Java application launcher (the "java" command) on HPSSGUI, the main class for the hpssgui utility.

F.40.4. Environment

The hpssgui uses a variety of methods to figure out the correct settings for running in the user's environment. These methods include: command line options, environment variable settings and SSM configuration file settings.

The command line options were discussed above and have the highest precedence (override environment variables and configuration variable settings).

For just the hpssgui.pl script, the environment variables are picked up by the user's environment variable settings. If the hpssgui.pl script is running on the HPSS core server machine, then it will attempt to pickup HPSS environment variable settings by running hpss_set_env -all and set any options represented in that list to the values from hpss_set_env(7), overriding the defaults and not changing the script's environment. See the man page for hpss_set_env(7) for more information about how it picks up HPSS environment variable settings from the user's environment and the system's env.conf file.

The hpssgui.vbs script does not take into account environment variables; for the most part, the user won't have access to hpss_set_env(7) from their Microsoft Windows desktop, so this won't be an issue. However, this is a shortcoming, because it won't attempt to pickup HPSS environment variables that the user may have previously set in their Windows environment.

The mkhpss tool generates $HPSS_PATH_SSM/ssm.conf, the SSM configuration file, on the core server machine. This file must be copied and possibly tailored for client machines that want to communicate with the System Manager. See the FILES section below for more information about the SSM configuration file.

Due to the multitude of supported environments (different client platforms and different security mechanisms), a set of defaults are chosen if not set by the command line, environment variable, and/or SSM configuration file:

```
        authentication mechanism:    Kerberos
```

This means that when you run the hpssgui script without any command line options or any override environment/configuration variables set, it will look for:

```
        java binary location     $PATH

        SSM configuration file   ./ssm.conf, OR
                                 $HPSS_PATH_SSM/ssm.conf

        krb5 configuration file  ./krb5.conf, OR
                                 /etc/krb5.conf

        login configuration file login.conf in the hpss.jar file , OR
                                 ./login.conf

        hpss.jar file            ./hpss.jar, OR
```

```
$HPSS_SSM_CLASSPATH
```

This means that for the following configurations, no command line options are required:

1. Running the hpssgui.pl on the same machine as your HPSS core server machine using Kerberos authentication. Note: This is not the recommended way to run since this generally involves remote X communication which is usually slow and sluggish.
2. Running the hpssgui.pl or hpssgui.vbs on a desktop that was configured using the steps outlined in the HPSS Admin Guide, is within the firewall of the HPSS core server machine, is using Kerberos authentication, and has the java binary in $PATH.
3. So, to run the hpssgui client on a Linux box doing Kerberos authentication without the java binary in the client's $PATH, you would need to do one of the following:
   - set JAVA_BIN user environment variable, or
   - set JAVA_BIN in ssm.conf, or
   - use the -j command line option to specify the java binary path

For hpssgui.pl, in general, the precedence rule is:

1. command line
2. environment
3. SSM configuration file
4. defaults

However, the handling of the SSM configuration file name for hpssgui.pl has a slightly modified precedence:

1. command line
2. environment
3. ./ssm.conf
4. `hpss_set_env HPSS_PATH_SSM`/ssm.conf.

For hpssgui.vbs, which doesn't process environment nor hpss_set_env(7) values, in general, the precedence rule is:

1. command line
2. SSM configuration file
3. defaults

with a slightly modified precedence for the SSM configuration file name:

1. command line
2. ./ssm.conf

The following table illustrates the relationships between command line options, environment

variables and SSM configuration variables.

Command Line Argument	Environment Variable	SSM Configuration Variable
-A	HPSSSSM_UI_ALARM_RATE	HPSS_SSM_ALARM_RATE
-C	HPSS_PATH_SSM/login.conf	LOGIN_CONFIG
-D	none	HPSS_SSM_DATE_FORMAT
-F	none	HPSSGUI_LOOK_AND_FEEL
-G	HPSS_SSM_ALARMS_GET	HPSS_SSM_ALARMS_GET
-L	HPSSSSM_UI_LIST_RATE	HPSS_SSM_LIST_RATE
-M	HPSSSSM_UI_MO_RATE	HPSS_SSM_MO_RATE
-N	HPSS_SSM_ALARMS_DISPLAY	HPSS_SSM_ALARMS_DISPLAY
-P	HPSS_SSM_CLASSPATH	HPSS_SSM_CLASSPATH
-S	none	LOG_FILE
-T	none	none
-W	HPSSSSM_UI_WAIT_TIME	HPSS_SSM_WAIT_TIME
-b	none	HPSSGUI_METAL_BG
-c	HPSS_SSM_COUNTRY	HPSS_SSM_CNTRY_CODE
-d	none	HPSS_SSM_DEBUG
-f	HPSS_HELP_FILES_PATH	HPSS_SSM_HELP_FILES_PATH
-g	HPSS_HELP_URL_TYPE	HPSS_SSM_HELP_URL_TYPE
-h	HPSSGUI_SM_HOST_NAME	HPSS_SSM_SM_HOST_NAME
-i	HPSSGUI_USER_CFG_PATH	HPSS_SSM_USER_PREF_PATH
-j	JAVA_BIN	JAVA_BIN
-k	KRB5_CONFIG	KRB5_CONFIG
-l	HPSS_SSM_LANGUAGE	HPSS_SSM_LANG_CODE
-m	HPSS_PATH_SSM/ssm.conf	SSM_CONFIG
-n	HPSSGUI_SM_PORT_NUM	HPSS_SSM_SM_PORT_NUM
-p	none	HPSS_SSM_CLIENT_IP
-r	HPSSGUI_RPC_PROT_LEVEL	HPSS_SSM_RPC_PROT_LEVEL
-s	HPSS_SSM_SEC_MECH	HPSS_SSM_SEC_MECH
-u	HPSSGUI_UNIX_DEFAULT_REALM	HPSS_SSM_UNIX_REALM
-x	none	none
--property	none	none

F.40.5. Files

/var/hpss/ssm/ssm.conf

The HPSS SSM configuration file. See the HPSS Admin Guide for details about configuring the HPSS Server Authentication Mechanism(s).

The mkhpss tool generates $HPSS_PATH_SSM/ssm.conf.

Depending on the authentication mechanism (Kerberos or Unix), mkhpss will put slightly different contents in the SSM configuration file. In all cases, the following variables will be defined:

HPSS_SSM_SM_HOST_NAME *host SM is running on* HPSS_SSM_SM_PORT_NUM *port SM is running on* HPSS_SSM_RPC_PROT_LEVEL *RPC protection level used by SM* HPSS_SSM_SEC_MECH *security mechanism used for SM communication*

If Unix authentication is enabled, the following variable will also be defined:

HPSS_SSM_UNIX_REALM *fully qualified SM UNIX realm name*

For example, if a site had both UNIX and Kerberos authentication configured for the System Manager, the ssm.conf file generated by mkhpss might look like:

```
ssm.conf:
    HPSS_SSM_SM_HOST_NAME   acme
    HPSS_SSM_SM_PORT_NUM    536870913:1
    HPSS_SSM_RPC_PROT_LEVEL connect
    HPSS_SSM_SEC_MECH       unix
    HPSS_SSM_UNIX_REALM     acme.com
```

Users could then select their appropriate authentication mechanism by setting HPSS_SSM_SEC_MECH to *krb5* or *unix*.

A site may personalize each SSM client's SSM configuration file by adding additional variables. For a list of these variables, see the table in the ENVIRONMENT section above. For example, a site may want to define the JAVA_BIN variable (which will be different depending on the client platform). Or a user managing multiple HPSS systems, may want to define the HPSSGUI_LOOK_AND_FEEL differently to better differentiate between the different systems. Or maybe a particular user wants the HPSS_SSM_DATE_FORMAT to have the year first whereas another user prefers the month first. All of the SSM configuration settings can be overridden by using command line options, but many users prefer setting everything in the SSM configuration file and not having to type/pass in any command line options.

The SSM client scripts all abort if they cannot find an SSM configuration file following the precedence rules described in the ENVIRONMENT section above.

A template for this file is provided in: /opt/hpss/config/templates/ssm.conf.template

/var/hpss/ssm/login.conf

The HPSS SSM login configuration file.

The default for this file is stored in the hpss.jar file as login.conf. In most cases, it should not be necessary to override this file.

If there is a need to override options in this file please see the /opt/hpss/config/templates/login.conf.template file and use the -C command line option to specify an alternative file.

A template for this file is provided in: /opt/hpss/config/templates/login.conf.template

/etc/krb5.conf

Kerberos V5 configuration file. This file is only required if using Kerberos authentication for SSM. This file is created during the process of configuring Kerberos on the HPSS core server machine.

By default the hpssadm script will look for this file in the current directory unless an alternate location is specified (by the -k option) or unless UNIX authentication is specified (by the -m and -u options).

Note: The -u command line option will override the HPSS_SSM_SEC_MECH variable in the SSM configuration file.

A template for this file is provided in: /opt/hpss/config/templates/krb5.conf.template

/opt/hpss/bin/hpss.jar file

The HPSS SSM client java archive file. This file contains the java binary classes required for starting up SSM clients.

The SSM client scripts all require this file.

AUTHZACL db2 table

The HPSS Authorization ACL table. Every user authorized to use an SSM client must have an appropriate entry in this db2 table. The SSM System Manager will reject connections from users without entries in the table, or with invalid entries in the table (e.g. invalid permissions). The table is read by the System Manager when a user logs in; so whenever SSM users are added or deleted from the table, the System Manager does NOT need to be restarted for the changes to take effect.

Users are added or removed from the table using the hpssuser(7) tool. See the hpssuser(7) man page. Users are added differently depending on the authority granted to the user (admin or operator). See the "Using SSM" section of the HPSS Admin Guide for more information about SSM authority and the required ACL settings for SSM users in this table.

/opt/hpss/ssmhelp/* (optional)

SSM Help files. The SSM Help Files are an HTML version of the HPSS Admin Guide. Individual sections of these guides are available from the Help menus on the SSM windows.

To access help windows from the hpssgui, the Help Files must be accessible from each client machine. We recommend storing these files in a file system shared by the clients so that they don't need to be installed on every SSM client machine. By default, the hpssgui script looks for the help files in $HPSS_HELP_FILES_PATH. The default location is /opt/hpss/ssmhelp and can be overridden

by using the -f command line option.

Help files are distributed with HPSS or can be downloaded from the HPSS web site. They should be installed in the $HPSS_HELP_FILES_PATH location and/or the path specified by the -f command line option. Refer to the HPSS Admin Guide for instructions on how to install the help files.

F.40.6. Notes

Performance

Tuning the System Manager RPC Thread Pool and Request Queue sizes can improve the performance of both the System Manager and the hpssgui clients. It is not necessary, however, to do the tuning when bringing up SSM for the first time. In fact, it can be helpful to postpone the tuning until after the site has a chance to learn its own SSM usage patterns.

The lower the refresh rates (-M -L -A options) the more CPU the client will use. The defaults for these are 5.

The higher the wait time (-W), the longer the RPCs will wait in the System Manager for the data object (list, MO) to change. This helps throttle the replies back to the client. However, the longer the RPCs wait in the System Manager's client interface thread pool, the more likely the thread pool is to fill up and RPCs start getting put into the client interface request queue. Once the request queue fills up, the RPC layer will start rejecting RPCs because there is nowhere to put them.

To understand the interaction of the refresh rates with the wait time, let's consider an example. If a user runs the SSM GUI startup script with -M set to 5 and -W set to 10, then for each Managed Object opened, the polling frequency will be between 5 and 15 seconds; a poll could occur as frequently as every 5 seconds but will definitely occur at least every 15 seconds.

If no updates were occurring for a particular Managed Object, then the gui would poll the SM, the SM would wait 10 seconds for something to change and then return, and then the gui would wait another 5 seconds before polling the SM again. So a poll would happen every 15 seconds.

If changes were constantly occurring for this particular Managed Object, then the gui would poll the SM, the SM would return immediately because he already had new information in his cache, the gui would update the screen with the new information, and then the gui would wait 5 seconds before polling the SM again. So a poll would happen every 5 seconds.

If changes were occurring intermittently (more frequently than 10 second intervals), the gui would poll the SM. The SM would wait 10 seconds or until he got new information, whichever came first, and then return. The gui would update the screen with the new information, wait another 5 seconds, and poll again. So more than 5 but less than 15 seconds would elapse between polls.

The System Manager client interface RPC thread pool size is defined in the "Thread Pool Size" field on the System Manager's Interface Controls server configuration window. This is the maximum number of RPCs that can be active at any one time for the client interface (i.e. all the hpssgui and hpssadm clients). For the server RPC interface (connections to the SSM System Manager from other HPSS servers), this value is determined by the HPSS_SM_SRV_TPOOL_SIZE environment variable.

The System Manager client interface RPC request queue size is defined in the "Request Queue Size" field on the System Manager's Interface Controls server configuration window. This is the

maximum number of RPC requests from hpssgui and hpssadm clients which can be queued and waiting to become active. For the server RPC interface this value is determined by the HPSS_SM_SRV_QUEUE_SIZE environment variable.

Ideally, if the site runs many clients, the client interface RPC thread pool size should be as large as possible; the default is 100. Testing this value at 300 showed the System Manager memory size more than double on AIX from around 32MB to over 70MB. The larger RPC thread pool size makes the System Manager much more responsive to the many clients but also requires it to use more memory.

Experimentation shows that leaving the client interface RPC thread pool size at 100 and leaving the client interface RPC request queue size at its default (600) works pretty well. During further experimentation, setting the client interface RPC request queue Size to 1000 saw very little effect on memory usage; with 40 clients connected the client interface RPC request queue used never went above about 500, but the client interface RPC thread pool was constantly filled.

Avoid allowing the client interface RPC thread pool to become full. When this happens, new RPCs will be put into the client interface RPC request queue to wait for a free thread from the thread pool. This makes the client response appear slow, because each RPC request is having to wait its turn in the queue. To help mitigate this, when the thread pool is full, the System Manager notifies all the threads in the thread pool that are waiting on list updates to go ahead and return to the client as if they just timed out as normal. This could be as many as 15 threads per client that are awakened and told to return, which makes those threads free to do other work.

If the client interface RPC thread pool is still full (as it could be if, for example, there were 15 threads in the client interface RPC request queue that took over the 15 that were just released), then the System Manager sets the wait time for the new RPCs to 1 second rather than whatever the client requested. This way the RPC won’t try to hang around too long.

Realize that once the System Manager gets in this mode (constantly having a full client interface RPC thread pool and having to cut short the thread waits times), the System Manager starts working hard and the CPU usage will start to increase. If you close some windows and/or some clients things should start to stabilize again.

You can see that the System Manager client interface RPC thread pool is full by looking at the "Maximum Active/Queued RPCs" field in the Client column of the "RPC Interface Information" group in the System Manager Statistics window. If this number is greater than or equal to the corresponding client interface’s "Thread Pool Size" (default 100), then the thread pool was full at some time during the System Manager execution (although it may not be full currently).

If "Active RPCs" is equal to "Thread Pool Size" then the thread pool for the interface is currently full. "Active RPCs" should never be greater than "Thread Pool Size". When it reaches "Thread Pool Size" then the new RPCs will be queued and "Queued RPCs" become greater than 0.

If "Queued RPCs" is currently 0 then the thread pool is not full. If "Maximum Active/Queued RPCs" is greater than "Thread Pool Size" then the thread pool was full at one time.

Hopefully if the thread pool was ever full it didn’t stay full for long. When it gets full then the System Manager tries harder to clear them out before accepting new ones.

If the site runs with low refresh rates and lots of clients, the recommendation is to set the client interface RPC thread pool size to 150 or 200 and the client interface RPC request queue size to 1000 in the System Manager Server Configuration.

F.40.7. Related Information

See the Using SSM section for more information on installing, managing and troubleshooting SSM.

F.41. hpsslimits(7)

F.41.1. Name

hpsslimits - Get core server limits for a subsystem

F.41.2. Syntax

hpsslimits [*OPTION*]

F.41.3. Description

This program prints core server limits. The limits by default are retrieved for subsystem 1, but can be retrieved for any subsystem using the -i option.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

See the HPSS Admin Guide for a description of fields in this output.

F.41.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

-i, --subsys
Subsystem ID (default=1)

F.41.5. Example

```
% /opt/hpss/bin/hpsslimits
        Max Open Bitfiles =              1001600
        Cur Open Bitfiles =                    0
            Max Active IO =                  480
            Cur Active IO =                    0
       Max Active Copy IO =                  115
       Cur Active Copy IO =                    0
          Max Connections =                  500
          Cur Connections =                    3
        Tape Queue Method =               Offset
    Tape Queue Active Set =                 1024
     Allow Any Characters =                 true
         Trashcan Enabled =                false
 Trashcan Unlink Eligible =               864000
           XML Size Limit =               131072
       XML Request Limit =               204800
  UDA Search Object Limit =                 4096
     UDA Search XML Limit =               131072
 UDA Search XML/Obj Limit =                 2048

% /opt/hpss/bin/hpsslimits -i 1
        Max Open Bitfiles =              1001600
        Cur Open Bitfiles =                    0
            Max Active IO =                  480
            Cur Active IO =                    0
       Max Active Copy IO =                  115
       Cur Active Copy IO =                    0
          Max Connections =                  500
          Cur Connections =                    3
        Tape Queue Method =               Offset
    Tape Queue Active Set =                 1024
     Allow Any Characters =                 true
         Trashcan Enabled =                false
 Trashcan Unlink Eligible =               864000
           XML Size Limit =               131072
       XML Request Limit =               204800
  UDA Search Object Limit =                 4096
     UDA Search XML Limit =               131072
 UDA Search XML/Obj Limit =                 2048
```

F.42. hpssln(7)

F.42.1. Name

hpssln - Create links (or symbolic links) in HPSS

F.42.2. Syntax

hpssln [*OPTION*] *TARGET LINK_NAME*

F.42.3. Description

This program creates a new link named *LINK_NAME* from the HPSS file, *TARGET*. The link can be either a hard link or symbolic link. By default, a hard link is created.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.42.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

-s, --symbolic
Symbolic link

F.42.5. Example

Example 1: Create a link in HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssln /path/to/file /path/to/link
```

F.43. hpss_license(7)

F.43.1. Name

hpss_license - Show HPSS license

F.43.2. Syntax

hpss_license show --license *license_path*
[-h] [-?]

F.43.3. Description

Display the current HPSS license file.

F.43.4. Parameters

--license *license_path*

Path to the HPSS license file.

-h, --help, -?

Display usage information.

F.43.5. Examples

To show the license:

```
/opt/hpss/bin/hpss_license show --license /var/hpss/etc/license.json
License Information:
  License ID: 9ec9fa6b-2e97-49a6-aff4-f4ae23c7761f
  Customer:   ACME
  Product:    hpss
  Issued:     2026-01-07 21:44:27 UTC
  Expires:    2030-01-01 00:00:00 UTC
  Grace:      30 days

Status: VALID
  License valid (1400 days remaining)
```

F.44. hpss_log_archive(7)

F.44.1. Name

hpss_log_archive - HPSS log archival tool

F.44.2. Syntax

hpss_log_archive -f *file-name*
[-a *auth-type*] [-l] [-p *user*] [-k] [-t *keytab*]
[-L *log-archive-path*] [-F *family-id*] [-c *cos-id*]
[-b *block size*] [-s] [-S *syslog-time-format*]
[-R] [-h] [-n] [-y] [-?]

F.44.3. Description

hpss_log_archive is a utility which copies local HPSS log files into HPSS. This allows HPSS log files to be archived so that they may be retrieved at some later date for diagnostic or information gathering purposes.

hpss_log_archive first opens the local log file in order to determine the start log time contained in the file. By default, it expects the syslog time format to be of the form "%b %d %H:%M:%S". The end time is taken from the file's last modified time. It then uses that information to construct the HPSS log path, which is of the form:

/log-archive-path/year/month/day/filename.YYYY.MMDD.HHMMSS.YYYY.MMDD.HHMMSS

If the start log time cannot be identified, then the log path will be of the form:

/log-archive-path/year/month/day/filename.YYYY.MMDD.HHMMSS

and will only use the last modified time.

If the --yesterday [-y] parameter is used, the log path will use the day before the last modified time instead of the last modified time.

If the --notimestamp [-n] parameter is used, the log path will use the last modified time, but the file name will not append any date/time stamp. The format will be:

/log-archive-path/year/month/day/filename

Next, it transfers the log file into HPSS. The -s option will log the exit status via syslog indicating whether the transfer was successful or not.

hpss_log_archive allows the caller to log in as a specific user, specify the log archive base path, and select the archived HPSS log file's COS and File Family.

hpss_log_archive also runs in a *recovery* mode with -R. In this mode, there is no error if the log file doesn't exist. It is treated as a success condition. This option can be used with a prerotate step in a logrotate definition to ensure that the previous log was archived before rolling over to the next file.

F.44.4. Options

-f *file-name*

Name of log file to archive.

-L *log-archive-path*

Log archive base path. Default: /log

-c *cos-id*

COS ID for the log archive file.

-b *block-size*

The block size to use. The default value is 100 MB. The block size may also be specified by the environment setting HPSS_LOG_BLKSIZE. The command line argument overrides the environment setting. A unit value may be specified after the numeric value. Valid unit values include: K or KB = Kilobyte
M or MB = Megabyte
G or GB = Gigabyte

-F *family-id*

File Family ID for the log archive file.

-s

(defunct) Log the exit status of this tool to syslog. NOTE: The tool always logs the exit status to HPSS

-R

Run in recovery mode. In this mode, the input log file not existing is not an error condition. This is meant to help recover from prior archive failures.

-S *syslog-time-format*

Start log timestamp format. See strptime(3). Default: "%b %d %H:%M:%S"

-a *auth-type*

Used to specify the authentication type to use. Supported values include:

1. unix - Simple Unix authentication
2. krb5 - Kerberos

-l

Used to create a new login credential. Only valid for *unix* authentication.

-p *user*

Used to specify the user principal name.

-k

Used to specify login using a keytab file.

-t *keytab*

Used to specify the keytab file path.

-n

Do not include date/time stamp in log archive file name.

-y

Archive the log to path using day before last modified date of the log.

-h,-?

Display help.

F.44.5. Example

```
$ ./bin/hpss_log_archive -f /var/hpss/log/HPSS.log
```

F.45. hpss_log_formatter(7)

F.45.1. Name

hpss_log_formatter - HPSS log formatter tool

F.45.2. Syntax

hpss_log_formatter [-h] [-f *FILE* [*FILE* ...]] [-s | -i] [-j]

F.45.3. Description

hpss_log_formatter is a utility that formats an HPSS syslog file and prints the contents in a more human readable format.

F.45.4. Options

-h, --help

Print a short help page describing the options available in hpss_log_formatter and exit.

-f, --file

syslog file name to format

-s, --strict

Enable strict parsing. This will raise an exception when we encounter an unprocessable line the in syslog input. Cannot be used with --ignore.

-i, --ignore

Ignore unprocessable records. This will not print anything when a record cannot be parsed. Cannot be used with --strict.

-j, --json

Output in JSON Lines format. This option cannot be used when reading from stdin.

F.45.5. Examples

```
$ ./bin/hpss_log_formatter -f /var/hpss/log/HPSS.log
Mar 25 11:17:38.045303 hpss_mvr_tcp(HPSS)144507::
Msg type: TRACE
Server:   Mover Disk (hpss-mvr)@hpss-mvr
Function: process_requests
Error:    0
MsgID:    MOVR0000
ReqID:    12103460669370363098
Payload:    Request ID = 12103460669370363098
            Function = READ
            Flags = REPLY_WHEN_READY
            SrcDescLength = 1
```

```
              SinkDescLength = 0
              REQSPEC: Subfunction = INVALID (0)
              REQSPEC: Count = 0
              REQSPEC: Argument = 0
              REQSPEC: Device ID = 0
              REQSPEC: Selection Flags = 0x00000000.0x00000000
              REQSPEC: Info type = REPLY_ADDR
              REQSPEC: ADDR: Addr type = NET:
              REQSPEC: ADDR: NET: Transfer ID = 0xa7f8215a16bb6cda
              REQSPEC: ADDR: NET: SockAddr.addr = 10.41.13.122
              REQSPEC: ADDR: NET: SockAddr.port = 43851
              REQSPEC: ADDR: NET: Offset = 0
              SRC:[0] Flags = HOLD_RESOURCES
              SRC:[0] Offset = 0
              SRC:[0] Length = 82023782
              SRC:[0] Shared Device Info = 0x0
              SRC:[0] Addr type = DEVICE:
              SRC:[0] DEVICE: Flags = HPSS
              SRC:[0] DEVICE: Device ID = 14
              SRC:[0] DEVICE: Block size = 4096
              SRC:[0] DEVICE: Blocks between TMs = 0
              SRC:[0] DEVICE: Volume ID = 00001400
              SRC:[0] DEVICE: POSITION: Flags = 0
              SRC:[0] DEVICE: POSITION: Granularity = 0
              SRC:[0] DEVICE: POSITION: RELATIVE: Section = 0
              SRC:[0] DEVICE: POSITION: RELATIVE: Offset = 134217728
              SRC:[0] DEVICE: POSITION: Absolute = 0x00000000 (INVALID)
              SRC:[0] DEVICE: POSITION: SSEG: RELATIVE: Section = 0
              SRC:[0] DEVICE: POSITION: SSEG: RELATIVE: Offset = 0
              SRC:[0] DEVICE: POSITION: SSEG: Absolute = 0x00000000 (INVALID)
              SRC:[0] Server defined = 0x0000000000000000
```

```
$ cat /var/hpss/log/HPSS.log | ./bin/hpss_log_formatter
Mar 25 11:17:38.045303 hpss_mvr_tcp(HPSS)144507::
Msg type: TRACE
Server:   Mover Disk (hpss-mvr)@hpss-mvr
Function: process_requests
Error:    0
MsgID:    MOVR0000
ReqID:    12103460669370363098
Payload:    Request ID = 12103460669370363098
            Function = READ
            Flags = REPLY_WHEN_READY
            SrcDescLength = 1
            SinkDescLength = 0
            REQSPEC: Subfunction = INVALID (0)
            REQSPEC: Count = 0
            REQSPEC: Argument = 0
            REQSPEC: Device ID = 0
            REQSPEC: Selection Flags = 0x00000000.0x00000000
```

```
REQSPEC: Info type = REPLY_ADDR
REQSPEC: ADDR: Addr type = NET:
REQSPEC: ADDR: NET: Transfer ID = 0xa7f8215a16bb6cda
REQSPEC: ADDR: NET: SockAddr.addr = 10.41.13.122
REQSPEC: ADDR: NET: SockAddr.port = 43851
REQSPEC: ADDR: NET: Offset = 0
SRC:[0] Flags = HOLD_RESOURCES
SRC:[0] Offset = 0
SRC:[0] Length = 82023782
SRC:[0] Shared Device Info = 0x0
SRC:[0] Addr type = DEVICE:
SRC:[0] DEVICE: Flags = HPSS
SRC:[0] DEVICE: Device ID = 14
SRC:[0] DEVICE: Block size = 4096
SRC:[0] DEVICE: Blocks between TMs = 0
SRC:[0] DEVICE: Volume ID = 00001400
SRC:[0] DEVICE: POSITION: Flags = 0
SRC:[0] DEVICE: POSITION: Granularity = 0
SRC:[0] DEVICE: POSITION: RELATIVE: Section = 0
SRC:[0] DEVICE: POSITION: RELATIVE: Offset = 134217728
SRC:[0] DEVICE: POSITION: Absolute = 0x00000000 (INVALID)
SRC:[0] DEVICE: POSITION: SSEG: RELATIVE: Section = 0
SRC:[0] DEVICE: POSITION: SSEG: RELATIVE: Offset = 0
SRC:[0] DEVICE: POSITION: SSEG: Absolute = 0x00000000 (INVALID)
SRC:[0] Server defined = 0x0000000000000000
```

F.46. hpssls(7)

F.46.1. Name

hpssls - List HPSS files

F.46.2. Syntax

hpssls [*OPTION*]... *FILE*

F.46.3. Description

This program lists HPSS files and directories.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

Since some short flags are used for multiple purposes, those necessary for authentication must be listed first while those that control the copy are listed second. The first flag that is unique to copying will make the rest of the flags be understood as copy options. One may also force following options to be copy options by using the — flag before the first copy option. The full-length flags will always be unambiguous, but the auth flags must still be in the first group.

F.46.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

-l, --long

Long listing

-a, --all

Show hidden files

-R, --recurse

Recursively process directories

-Z, --epoch

Print times as seconds since epoch

-H, --hpssattrs

Print HPSS attributes (COS, Account, Family)

-S, --status

Display summary statistics

F.46.5. Example

Example 1: Long listing of a directory in HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssls -- -l /tmp/testdir
-rw-rw-r--   1 0       0            1024 Mar 24 09:34 /tmp/testdir/file1
-rw-rw-r--   1 0       0            2048 Mar 24 09:34 /tmp/testdir/file3
```

Example 2: Long listing of all files in a directory in HPSS with user/pass

```
# hpssls -l -- -a -l /tmp/testdir
```

F.47. hpssmkdir(7)

F.47.1. Name

hpssmkdir - Make directories in HPSS

F.47.2. Syntax

hpssmkdir [*OPTION*] *DIRECTORY*...

F.47.3. Description

This program creates directories in HPSS.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.47.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

-p,--parents
Create parent directories

F.47.5. Example

Example 1: Create a directory in HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssmkdir /tmp/path/to/testdir
```

Example 2: Create a directory in HPSS, and all parent directories

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssmkdir -p /tmp/path/to/testdir
```

F.48. hpssmsg(7)

F.48.1. Name

hpssmsg - Generate an HPSS log message

F.48.2. Syntax

hpssmsg [-s *severity level*] [-t *log type*] [-i *message id*]
[-r *return code*] [-q *request ID*] [-p *program name*]
[-f *function name*] [-h] {-m *message* | *message*}

F.48.3. Description

The hpssmsg utility generates a log message by writing to the syslog in the format expected by HPSS. Test programs can use it to log status and completion results. These messages plus the messages logged by HPSS can be used to analyze and troubleshoot issues. It can also be used to generate messages which are shown in the SSM alarms and events window, notifying other administrators, for example, when a server is about to be restarted. It can also be used by remote HPSS users to report HPSSFS-FUSE, PFTP, or other problems they are encountering.

The user is required to specify a message and may specify one or more optional values, including the severity level, log type, message ID, return code, request ID, program name, and function. All of the optional values have default values. The combination of the user-supplied message and the optional values are used in the format of the message written to the syslog.

The "-t" option defines the type of HPSS log message. Note that, only "alarm" and "event" messages will be shown on the HPSS Storage System Manager display. The "-s" option defines the severity of the message.



Improper use of this utility could overburden the HPSS logging facility as well as the SSM. Administrators may wish to restrict access to this tool.

F.48.4. Parameters

Required command-line arguments:

-m,--message *message*

You may also specify the message at the end without specifying one of these arguments

Optional command-line arguments:

-s,--severity *severity level* (defaults to *none*)

Where *severity level* is one of: none, warning, minor, major, critical, indeterminate, cleared. Note, for all log types except alarm, "none" is the only valid severity level.

-t,--log_type *log type* (defaults to *event*)

Where *log type* is one of: alarm, event, request, security, accounting, debug, trace, info

-i,--message_id *message id* (defaults to *USER0001*)

Where *message id* is a string nine characters or less that should be in the format <COMPONENT><#>. The value entered is converted to uppercase

-r,--return_code *return code* (defaults to *0*)

Where *return code* is an integer

-q,--request_id *request id* (defaults to 32 '0's)

Where *request id* is a 32-character hexadecimal value. The value entered is converted to lowercase

-p,--program_name *program name* (defaults to *hpssmsg*)

Where *program name* is a string value corresponding to the name of the program

-f,--function *function name* (defaults to *N/A*)

Where *function name* is a string representing the location in the code generating the message. This is typically the file name, function name, and line number

-h,--help

Displays usage information

F.48.5. Examples

```
     % hpssmsg -t debug -s none Beginning stress test on node `hostname`

     % hpssmsg -t alarm -s critical -i USER0002 -p watcher -m "NFS daemon appears to
be hung"

     % hpssmsg -t info -s none -i USER0003 -p hpssadm_hs_info -r 1 -q
bc508254d4ad3c4bb195c75ed076ad24 -m "Error capturing statistics: unable to create
input pipeline"

     % hpssmsg -t alarm Tracy will be recycling the PVL over the next 30 minutes
```

F.48.6. See Also

hpss_logarchive

F.49. hpss_pftpd(7)

F.49.1. Name

hpss_pftpd - HPSS Parallel FTP (PFTP) Daemon

F.49.2. Synopsis



The HPSS Parallel FTP daemon is normally started by the inetd daemon (AIX) or the xinetd daemon (Linux). TCP wrapper applications may also be specified (NOT provided by HPSS.)

hpss_pftpd [-c *HPSSAltFileName*]

F.49.3. Description

The HPSS PFTP daemon is the HPSS parallel version of the DARPA Internet File Transfer Protocol (FTP) server process. The HPSS PFTP daemon uses the Transmission Control Protocol (TCP) to listen at the port specified with the FTP command service specification in the /etc/services file.

The Credential-based (Generic Security Service, GSS) protocol supports cross-cell authentication. With cross-cell authentication the client user's home directory is read from the appropriate authorization file (either LDAP or password/group files).

F.49.4. Options

-c *HPSSAltFileName*

Alternate name for the initialization file (default: HPSS.conf).

F.49.5. Supported File Transfer Protocol Requests

The pftpd daemon currently supports the following FTP requests:

ABOR

Terminates previous command.

ACCT

Specifies account (ignored).

ALLO

Allocates storage (vacuously).

APPE

Appends to a file.

CDUP

Changes to the parent directory of the current working directory.

CWD

Changes working directory.

DELE

Deletes a file.

EPRT

Specifies a data connection port (IPv6).

EPSV

Prepares for passive transfers (IPv6).

FEAT

Asks the server for a list of supported features.

HELP

Gives help information.

IDLE

Sets idler time

LIST

Gives list files in a directory.

MKD

Makes a directory.

MDTM

Shows last modification time of file.

MLSD

Gives a list of files in directory.

MODE

Specifies data transfer mode.

NLST

Gives a name list of files in directory.

NOOP

Does nothing.

OPTS

Allows the client to configure options for specific commands.

PASS

Specifies a password.

PASV

Prepares for passive transfers (IPv4).

PORT

Specifies a data connection port (IPv4).

PWD

Prints the current working directory.

QUIT

Terminates session.

RETR

Retrieves a file.

RMD

Removes a directory.

RNFR

Specifies rename-from file name.

RNTO

Specifies rename-to file name.

SITE

Supported commands:

CHMOD

Changes mode of a file

HELP

Gives help information

UMASK

Changes umask

SIZE

Returns size of current file. (Binary size)

STAT

Returns the status of the server.

STOR

Stores a file.

STOU

Stores a file using a unique file name.

STRU

Specifies the structure of data transfer as a file structure.

SYST

Shows operating system type of server system.

TYPE

Specifies data transfer type with the Type parameter.

USER

Specifies user name.

F.49.6. HPSS Supported FTP Extensions

AUTH

Authentication type.

ADAT

Authentication data.

LFRTR

Local File Retrieve.

LFSTO

Local File Store.

PBSZ

Parallel buffer size.

PCLO

Parallel close.

POPN

Parallel open.

PPOR

Parallel port.

PRLL

Supports Parallel Transfer Operations.

PROT

Protection level.

PRTR

Parallel retrieve.

PSTO

Parallel store.

SAN3P

Enable/disable SAN3P.

SITE

Supported commands:

ACL

Change/Add ACLs to HPSS Objects

CHGID

Change group id.

CHGRP

Change group.

CHOWN

Change owner.

CHUID

Change user id.

GETFAM

Get file family of file

RPTI

Set Transfer Agent report interval.

SCKT

Socket address for Transfer Agent

SETCOS

Set class of service.

SETFAM

Set file family

STAGE

Stage file.

SYMLINK

Symbolic link.

TAC

Get number of transfer agents to use.

WAIT

Wait-on-stage time.

F.49.7. Building a Configuration File

A configuration file will be used to specify options for hpss_pftpd. The default configuration file is /var/hpss/etc/HPSS.conf. The user can override the path to this file with the HPSS_CFG_FILE_PATH environment variable or by using the -c option. It is the Administrator's responsibility to correctly configure and maintain the HPSS.conf file as well as implementing the multi- noded components on all remote systems.

F.49.8. Files

/etc/inetd.conf(AIX)

/etc/xinetd.d/hpssftp(Linux)

/etc/services

/etc/syslog.conf

/var/hpss/etc/HPSS.conf

/var/hpss/etc/authz.conf

/var/hpss/ftp/adm/hpss_ftpd.log

/var/hpss/ftp/adm/xferlog

/var/hpss/etc/ftpaccess

/var/hpss/etc/ftpbanner

/var/hpss/etc/passwd

/var/hpss/etc/shadow

/var/hpss/etc/group

/var/hpss/etc/ftpusers

Appropriate Shared Libraries.

F.49.9. Environment

HPSS_CFG_FILE_PATH Alternate path to HPSS.conf

F.49.10. Examples

Templates for configuring /etc/services, inetd.conf(AIX), and /etc/xinetd.d/hpssftp(Linux) are located in /opt/hpss/config/templates/system_files.

F.49.11. See Also

ftp,hpss_ftp,pftp_client,syslog,xinetd,HPSS.conf

HPSS User's Guide

HPSS Admin Guide

F.49.12. Author

Tim Preston, Sandia National Labs, Albuquerque, NM

F.50. hpsspurge(7)

F.50.1. Name

hpsspurge - Purge files in HPSS

F.50.2. Syntax

hpsspurge *FILE...*

F.50.3. Description

This program purges one or more files in HPSS from level 0.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.50.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.50.5. Example

Example 1: Purge a file in HPSS

```
# export HPSS_CLI_USER=myuser
```

```
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsspurge /tmp/path/to/file
Purged '/tmp/path/to/file'
```

F.51. hpsspurgelock(7)

F.51.1. Name

hpsspurgelock - Purge lock or purge unlock HPSS files

F.51.2. Syntax

hpsspurgelock [*OPTION*] *FILE...*

F.51.3. Description

This program allows users to purge lock or purge unlock HPSS files.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.51.4. Options

-h,--help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

-u,--purge-unlock
Purge unlock the files specified. If not specified, files will be purge locked.

F.51.5. Example

Example 1: Purge lock HPSS files

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsspurgelock /tmp/file.1 /tmp/file.2
Purge locked file '/tmp/file.1'
Purge locked file '/tmp/file.2'
```

Example 2: Purge unlock HPSS files

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsspurgelock -u /tmp/file.1 /tmp/file.2
Purge unlocked file '/tmp/file.1'
Purge unlocked file '/tmp/file.2'
```

F.52. hpss_recurse(7)

F.52.1. Name

hpss_recurse - Traverse down a directory tree in HPSS and run a callback on each file or directory in a multi-threaded manner

F.52.2. Syntax

hpss_recurse [-a *auth type*] [-l] [-k] [-t *keytab_path*]
[-p *principal*] [-n *num_threads*] [-v] [--logfile *logfile*]
-b *directory path* -c *callback path* -C *callback args*
[-h]

F.52.3. Description

This tool traverses a directory tree with a breadth first search algorithm in a multi-threaded manner from the given base directory.

A user-supplied callback with user-supplied parameters is run on every file and directory that is traversed from the base directory.

If an error is encountered as the tool is processing a file or directory, that error is logged, and the tool continues execution. It is up to the user to fix the cause of the error and run their callback on the failed path again. The error message will always include the path of the file or directory in HPSS that encountered the error. This will be logged as path:<string> and will appear at the bottom of the traceback in the following format.

<Error Type>: path:<path> <error message>

The traceback and error message should be descriptive enough for the user to find the source of the issue.

F.52.4. Options

Required command-line arguments:

-b,--basedir

directory path Base directory in HPSS to traverse down

-c,--callback

callback path Path to Python file that implements callback_func, get_callback, and verify_and_convert_args_str functions from the hpss_recurse_cb_interface.py interface to be imported into this script

-C,--cbargs

callback arguments Arguments for the callback provided as a string.

Optional command-line arguments:

-a,--auth-type

The type of authentication to use, either "krb5" or "unix". If not provided, will be determined based on HPSS_CLIENT_AUTHN_MECH environment variable

-l

Get a new login credential

-k,--keytab

Use a keytab

-t,--keytab-file

Path of the keytab file to use

-p,--login-principal

User to login to HPSS as

-n,--num-threads

Number of threads to use for the thread pool

-v,--verbose

Log output in verbose mode, which will log every callback success. Without verbose mode, only failures are logged.

--logfile *logfile*

Path of file to log output to. Without --logfile, output is logged to stderr.

-h,--help

Displays usage information

F.52.5. Examples

This will chown every file and directory in HPSS directory "/small_dir", including "/small_dir", to the user with uid 300 and the group with gid 300. Will set every file to permissions "600" (rw-------) and every directory to permissions "700" (rwx------). Login to HPSS as user principal "hpss" and use the keytab "/var/hpss/etc/hpss.unix.keytab".

```
% hpss_recurse -b /small_dir -k -t /var/hpss/etc/hpss.unix.keytab -p hpss
-c /opt/hpss/lib/py-packages/chmod_chown_callback.py
-C "-o 300 -g 300"
```

This will chown every file and directory in HPSS directory "/small_dir", including "/small_dir", to the user with uid 300 and the group with gid 300. Will set every file to permissions "567" (r-xrw-rwx) and every directory to permissions "123" (--x-w—wx). Login to HPSS as user principal "hpss" and use the keytab "/var/hpss/etc/hpss.unix.keytab".

```
% hpss_recurse -b /small_dir -k -t /var/hpss/etc/hpss.unix.keytab -p hpss
-c /opt/hpss/lib/py-packages/chmod_chown_callback.py
```

```
-C "--owner 300 --group 300 -f 567 -d 123"
```

Example of implementing the callback. See the pydocs of the hpss_recurse_cb_interface.py file for more details.

The hpss_recurse_cb_interface interface has four functions that must be implemented by a callback, and a global class_name parameter that must be set. The four functions that must be implemented are

1. **__init__(self)** This functions should setup a self.parser attribute that uses the argparse module to handle arguments to this callback and print out the usage and the help.
2. **add_arguments(self)** Adds arguments to the self.parser attribute
3. **get_callback(self, args)** Retrieves callback method to be called by hpss_recurse.py script
4. **callback_func(self, path, handle, **kwargs)** The callback method implemented

The global class_name parameter should be set to the name of the class that is implementing hpss_recurse_cb_interface.

As an example, say you want to implement a callback print_path_callback that prints the path of a traversed file or directory. It takes a single optional argument, "-f", that if set will print the full path for files. If not set, just the file name will be printed for files. The tool will always print the full path for directories. This file will be named print_path_callback.py and will be placed in this location /opt/hpss/lib/py-packages/hpss_recurse_callbacks

It is expected all callbacks for this tool to go into that directory.

So that file should look like this.

```
    import os
    import sys
    import logging
    from argparse import ArgumentParser, HelpFormatter
    from functools import partial
    # import the hpss extension module
    import hpss.clapi as hpss

    expected_tool_path = \
        f'{os.sep}opt{os.sep}hpss{os.sep}lib{os.sep}' + \
        f'py-packages{os.sep}hpss_recurse_callbacks'

    if expected_tool_path not in sys.path:
        sys.path.append(expected_tool_path)

    from hpss_recurse_cb_interface import hpss_recurse_cb_interface

    BOLD_TEXT = "\033[1m"
    END = "\033[0m"
    class_name = "print_path_callback"
```

```
 def __init__(self):
     """!
     @brief constructor, creates self.parser class attribute
     """
     self.parser = \
         ArgumentParser(prog=f"{__file__}",
                        formatter_class=HelpFormatter,
                        description="{}DESCRIPTION{} callback to "
                        "print the path of every file and directory in the "
                        "tree. Use following arguments for -C option of "
                        "hpss_recurse.py".format(BOLD_TEXT, END))

def get_callback(self, args):
     """!
     @brief Returns callback function to be called later. Callback
     function must be called with the path and handle parameters.

     @param args: dict, arguments to hpss_recurse.py and this callback

     @returns method, Only need to call returned method with
         path and handle parameters, due to 'partial' module
     """
     return partial(self.callback_func, full_path=args["full_path"])

 def callback_func(self, path, handle, **kwargs):
     """!
     @brief Callback function to be called by hpss_recurse.py. prints
     full path of directories. Prints full path of files if the
     keyword argument full_path is given, otherwise prints relative path of
     file.

     @param path: str, full path of file or directory to print
     @param handle: obj, handle of file or directory

     @param kwargs: keyword args to use with callback function. Keywords
         are:
     @verbatim
         full_path: bool or None, Do we want to print full path of file?
             Is None if "-f" argument was not provided.
     @endverbatim
     """
     # handling directory or have used optional "-f" option.
     if handle.Type == hpss.NS_OBJECT_TYPE_DIRECTORY or \
             kwargs.get("full_path"):
         print(path)
     # print just the file name
     else:
         print(path.rsplit("/", 1)[1])
```

```
    def add_arguments(self):
        """!
        @brief add arguments used by callback to the argument parser.
        """
        self.parser.add_argument("-f",  action="store_true", dest="full_path",
                                 help="If set, will print out the full path "
                                 "of files")
```

So to call that print_path_callback.py as our callback with the "-f" option on the HPSS directory "/small_dir":

```
    % hpss_recurse -c <path to print_path_callback.py> -C="-f" -b "/small_dir"
```

So to call that print_path_callback.py as our callback without the "-f" option on the HPSS directory "/small_dir":

```
    % hpss_recurse -c <path to print_path_callback.py> -b "/small_dir"
```

F.53. hpssrm(7)

F.53.1. Name

hpssrm - Delete HPSS files

F.53.2. Syntax

hpssrm *FILE...*

F.53.3. Description

This program deletes HPSS files. Note that directories cannot be deleted with this tool (use hpssrmdir).

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.53.4. Options

-h,--help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.53.5. Example

Example 1: Delete a file in HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssrm /tmp/path/to/file
```

F.54. hpssrmdir(7)

F.54.1. Name

hpssrmdir - Delete HPSS directories

F.54.2. Syntax

hpssrmdir *DIRECTORY...*

F.54.3. Description

This program deletes HPSS directories. A directory must be empty in order to be deleted. Note that files cannot be deleted with this tool (use hpssrm).

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.54.4. Options

-h,--help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.54.5. Example

Example 1: Delete a directory in HPSS

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssrmdir /tmp/path/to/dir
```

F.55. hpss_server_acl(7)

F.55.1. Name

hpss_server_acl - Modify server security ACLs

F.55.2. Syntax

hpss_server_acl

F.55.3. Description

This utility allows the user to view and edit the security ACLs for the HPSS servers. It accepts a sequence of commands on standard input. The user running this utility must have authority to access the HPSS metadata, where the security ACLs are stored. Each HPSS server has at least one RPC interface, and each interface has an ACL.

F.55.4. Commands

acl

The "acl" command is used to select the server interface whose ACL you wish to work on. It has several forms:

- acl -n "server name" [<interface>]
- acl -u <server uuid> [<interface>]
- acl -t <server type> [<interface>]

These forms select a server by its descriptive name, uuid, or type, respectively. See the section INTERFACE SELECTION for details of what to pass for the -t option and for the interface selection arguments.

acl -m

This form allows the ACL to be selected from a menu.

acl

This form displays information about which interface of which server is currently selected.

show

The "show" command displays the contents of the current ACL.

default

The "default" command sets the ACLs for each of the current server's interfaces to their default values.

clear

The "clear" command is used to remove all entries from the current ACL.

add *type args perms*

The "add" command is used to add an entry or permissions to the current ACL. See the ACL SYTNAX section below for a description of the form of the arguments. If the specified type of entry does not already exist in the ACL, it is added with the specified permissions. If the entry does already exist, the specified permissions are added to it.

del [*type args* [*perms*]]

The "del" command is used to remove an entry or permissions from the current ACL. See the ACL SYTNAX section below for a description of the form of the arguments. If no permissions are specified in the arguments, the entire entry specified is removed from the ACL. If permissions are specified, those permissions are removed from the specified entry, but the entry itself is not removed. If "del" is entered with no arguments, the entry to operate on id selected from a menu.

help

The "help" command displays a message with a summary of command syntax.

quit

The "quit" command exits the program.

F.55.5. ACL Syntax

Following is a description of the various ACL entry types and permissions that can be used with the "add" and "del" commands:

ACL entry types and args:

user *name* | *id*

For a user in the local realm.

foreign_user *realm* | *realmId name* | *id*

For a user in a foreign realm.

group *name* | *id*

For a group in the local realm.

foreign_group *realm* | *realmId name* | *id*

For a group in a foreign realm.

foreign_other *realm* | *realmId*

This applies to an entity of the specified foreign realm to which no other entry applies.

any_other

This applies to an entity to which no other entry applies.

In any entry where a *name* of *id* is used, either the name of the entity (user or group) or its numeric ID is acceptable.

In any entry where a *realm* or *realmId* is used either the name of the realm or its numeric ID is acceptable.

Permissions (a string consisting of any combination of the following in any order can be used: rwxcidt):

r=read	w=write	x=execute	c=control
i=insert	d=delete	t=test	

F.55.6. Interface Selection

These options are for use with various forms of the "acl" command to select the server interface to operate on:

-U *if uuid* [-v *version*]

This form selects the interface by its uuid.

-d "*if desc*" [-v *version*]

This form selects the interface by its descriptive name.

-T *standard if type*

This form selects the interface by one of the names listed below.

These are the standard server and interfaces types for use with the -t and -T options, respectively, of the "acl" command:

Server	**Interface Type**
CORE	core sspvl rtm acctval
DMG	dmgadmin dmgdm
GK	gk gkadmin aval rtm
HPSSD	hpssd
LOGC	logc
LOGD	logd
LS	ls lspeer
MNT	mnt
MPS	mps
MVR	mvr rtm
NFS	nfs
PVL	pvl
PVR	pvr
SSM	ssmclient ssmserver

F.55.7. Examples

Start a session with hpss_server_acl:

```
$ hpss_server_acl
```

Select the client interface of the SSM System Manager:

```
hsa> acl -t SSM -T ssmclient
```

Show the current contents of the ACL:

```
hsa> show
perms - type - ID (name) - realm ID (realm)
=========================================
rwxcidt - user - 0 (root) - 200075 (FIRE.CLEARLAKE.IBM.COM)
rwxcidt - user - 191 (client3) - 200075 (FIRE.CLEARLAKE.IBM.COM)
rwxcidt - user - 192 (client2) - 200075 (FIRE.CLEARLAKE.IBM.COM)
rwxcidt - user - 193 (client1) - 200075 (FIRE.CLEARLAKE.IBM.COM)
```

Add a new entry to the ACL:

```
hsa> add user client4 rwxcidt
```

Remove some permissions from an entry:

```
hsa> del user client1 wx
```

Show the current contents of the ACL:

```
hsa> show
perms - type - ID (name) - realm ID (realm)
=========================================
rwxcidt - user - 0 (root) - 200075 (FIRE.CLEARLAKE.IBM.COM)
rwxcidt - user - 191 (client3) - 200075 (FIRE.CLEARLAKE.IBM.COM)
rwxcidt - user - 192 (client2) - 200075 (FIRE.CLEARLAKE.IBM.COM)
r--cidt - user - 193 (client1) - 200075 (FIRE.CLEARLAKE.IBM.COM)
rwxcidt - user - 194 (client4) - 200075 (FIRE.CLEARLAKE.IBM.COM)
```

F.56. hpss_server_metrics(7)

F.56.1. Name

hpss_server_metrics - query the HPSS system for metrics from the HPSS servers.

F.56.2. Syntax

hpss_server_metrics -t *mover*|*storage*|*config*|*limits*|*statfs*|*core*|*drives*|*trash*|*all*
[-m *authentication mechanism*]
[-k *authenticator name*]
[-s *schema name*]
[-g *global database name*]
[-p *login principal*]
[-j]
[-r]
[-h] [-?]

F.56.3. Description

hpss_server_metrics reports on numerous metrics collected by the HPSS servers.

The metrics reported by <report type> are:

mover
: Reports the number of requests, bytes moved, and errors for each mover.

storage
: Reports a summary for all the storage on the system.

config
: Reports how many HPSS servers and devices are configured.

limits
: Reports the limits information for each core server.

statfs
: Reports the statfs information for each COS.

core
: Reports each core server's cache hit rate, frequency of API calls, and error rates.

trash
: Reports each core server's trash can statistics.

drives
: Reports the bytes read, bytes written, mounts, and errors for each device and drive.

all

Dumps all the report types.

F.56.4. Parameters

-t *type*

The report type. Multiple report types may be specified with a space between each.

-m *authentication mechanism*

The authentication mechanism to use. If not specified, the value of environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

-k *authenticator name*

The authenticator which is used to obtain credentials. If not specified, the value of environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

-s *schema name*

The schema name. If not specified, the value of environment variable HPSS_MM_SCHEMA_NAME will be used.

-g *global database name*

The global configuration database name. If not specified, the value of environment variable HPSS_GLOBAL_DB_NAME will be used.

-p *login principal*

The server principal name under which this program will run. If not specified, the value of environment variable HPSS_PRINCIPAL_SSM will be used.

-j

Output in JSON format.

-r

This option will reset any resettable stats for any reports selected. This includes the core server statistics, mover statistics, and device and drive statistics. NOTE: Device statistics for "Mounts since last maintaince" and "Drive error count" are not reset.

-?,-h

Display usage information.

F.56.5. Handling Unavailable Servers

When collecting metrics that require connecting to servers (such as `config`, `storage`, or `drives` report types), the tool will continue to report metrics for all available servers even if some are unavailable.

F.56.5.1. Behavior

- The tool will skip devices on unavailable movers and continue processing
- A warning message is printed to stderr for each unavailable server

- Available metrics are reported normally
- The tool exits with success status (0) even when some servers are unavailable

F.56.6. Example 1

```
    $ hpss_server_metrics -t statfs

    *** COS 1 Stats:

    Block size is 1048576
    Block count is 509950
    Free block count is 482282
    Free block available count is 482282
    File count is 2000
    Free file count is 2000
    File system id is 1
    File max name is 256
    File system name is Example
```

F.56.7. Example 2 - Unavailable Mover

```
$ hpss_server_metrics -t config
Warning: Unable to connect to mover Mover (node2) (ID 3), skipping its devices

** Configuration Stats:

        Total HPSS Servers:          10
        Total HPSS Subsystems:       1
        Total HPSS Nodes:            4
        Total Movers:                4
        Total Disk Devices:          6
        Total Tape Devices:          8
        Total Object Store Devices:  0
Completed
```

And the JSON output:

```
$ hpss_server_metrics -t config
{
 "ConfigMetrics": {
  "TotalHPSSServers": 10,
  "TotalSubsystems": 1,
  "TotalNodes": 4,
  "TotalMovers": 4,
  "TotalDiskDevices": 6,
  "TotalTapeDevices": 8,
  "TotalTapeMounts": 0,
```

```
   "TotalObjectStoreDevices": 0
  },
  "success": true,
  "UnavailableServers": {
   "Movers": [
    {
     "Id": 3,
     "Name": "Mover (node2)"
    }
   ]
  },
  "ErrorMessage": "Completed"
 }
```

The `success` field will still be set to `true` when any servers are unavailable, and the `UnavailableServers` section lists all movers that could not be contacted during the collection process. The `ErrorMessage` field will indicate what error occurred. The `success` field is only set to `false` when the tool failed in a way where it could not gather error information or could not run at all.

In this example, mover ID 3 was unavailable, so its devices were not included in the device counts, but all other metrics were collected successfully.

F.57. hpsssetcos(7)

F.57.1. Name

hpsssetcos - Set the COS of one or more HPSS files

F.57.2. Syntax

hpsssetcos [--stream-id *stream*] *COS FILE...*

F.57.3. Description

This program sets the COS ID on files in HPSS. This causes the files to be changed to the target COS ID in the background. Providing a COS of 0 causes a prior change cos operation on the file to be canceled.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

COS changes occur in background threads called "streams". The caller can set the stream to use for organizing COS change operations using the --stream-id parameter.

F.57.4. Options

-h,--help

Display usage information.

-s,--stream-id *stream*

COS change stream to use.

--authtype *type*

Authentication type

--keytab

Use keytab for auth

--ktpath *keytab-path*

Set keytab path

--newcred

Create new creds (implies password)

--principal *principal*

Use principal name

F.57.5. Example

Example 1: Set the COS on a file

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsssetcos 1 /tmp/path/to/testdir
```

Example 2: Set the COS on a file using stream ID 14

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsssetcos --stream-id 14 1 /tmp/path/to/testdir
```

F.58. hpss_set_env(7)

F.58.1. Name

hpss_set_env - query/set HPSS environment

F.58.2. Syntax

hpss_set_env [-all]

hpss_set_env [-def]

hpss_set_env [-set *ENVNAME=NEWVALUE*]

hpss_set_env [*ENVNAME* [*ENVNAME*...]]

F.58.3. Description

HPSS defines its own environment, separate from the user's process environment, and accessed through the hpss_set_env utility. The HPSS values can be set in the user's process environment, but by default they are not.

hpss_set_env can report or set the current HPSS environment. The option -all gets a complete list of the current HPSS environment. The option -def gets a complete list of default HPSS environment values. The option -set is used to update the HPSS environment override file, normally `/var/hpss/etc/env.conf`. Once a value is set in the override file, hpss_set_env will report the override value. Variable names listed on the command line will be reported with their values in the program's output.

hpss_set_env is designed to be run using a single option or operational mode at a time. For example, if **-all** and **-def** are specified together on the command line, the result may not be what is desired or expected.

In general, if more than one option is used on the command line, only the first one will be processed. If environment variable names are listed on the command line along with an option, the variables before the option will be reported, the option will be processed, and the program will exit without processing any command line arguments following the option.

Most HPSS utilities reside in the hpss bin directory, normally `/opt/hpss/bin`. hpss_set_env is an exception since it sits in `$HPSS_ROOT/config` (normally `/opt/hpss/config`).

The perl module hpss_env.pm calls hpss_set_env to do its work.

F.58.4. Examples

The following command lists all values in the current HPSS environment.

```
$ /opt/hpss/config/hpss_set_env -all
HPSS_ROOT                    ="/opt/hpss"
```

```
HPSS_HOST                    ="maryjo"
HPSS_NODE_TYPE               =""
HPSS_PATH_INSTALL            ="/opt/hpss"
HPSS_PATH_BIN                ="/opt/hpss/bin"
HPSS_PATH_MSG                ="/opt/hpss/msg/En_US"
...
```

The following command lists default values for the HPSS environment.

```
$ /opt/hpss/config/hpss_set_env -def
HPSS_ROOT                    ="/opt/hpss"
HPSS_HOST                    ="%H"
HPSS_NODE_TYPE               =""
HPSS_PATH_INSTALL            ="/opt/hpss"
HPSS_PATH_BIN                ="${HPSS_PATH_INSTALL}/bin"
HPSS_PATH_MSG                ="${HPSS_PATH_INSTALL}/msg/En_US"
...
```

The following command will update the env.conf file with the indicated setting so that future client sessions launched on the current machine will use the indicated port number.

```
$ /opt/hpss/config/hpss_set_env -set HPSSGUI_SM_PORT_NUM=536870943:1
```

The following command displays the values of the environment variables listed on the command line.

```
$ /opt/hpss/config/hpss_set_env HPSS_ROOT HPSS_PATH_VAR HPSS_USER
HPSS_ROOT="/opt/hpss"
HPSS_PATH_VAR="/var/hpss"
HPSS_USER="hpss"
```

F.58.5. Files

/var/hpss/etc/env.conf

HPSS environment overrides

F.58.6. See Also

hpss_env.pm

F.59. hpssstat(7)

F.59.1. Name

hpssstat - Stat an HPSS file

F.59.2. Syntax

hpssstat *FILE...*

F.59.3. Description

This program stats an HPSS file.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.59.4. Options

-h,--help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.59.5. Example

Example 1: Stat an HPSS file

```
# export HPSS_CLI_USER=myuser
```

```
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssstat /tmp/path/to/file
```

F.60. hpsssum(7)

F.60.1. Name

hpsssum - HPSS Checksum Tool

F.60.2. Syntax

hpsssum -c *hash-type* -f *file-name* -F *file-list-file* [-n *num-threads*]
[-L] [-a *auth-type*] [-l] [-p *user*] [-k] [-t *keytab*]
[-u *command*] [-U *command*] [-d] [-g] [-i *subsystem-id*]
[-b *buffer-size*] [-r *max-results*] [-v] [-h] [-s] [-R] [-S] [-N]

F.60.3. Description

hpsssum is a utility which uses the HPSS Checksum client API functions to provide checksums of both local and HPSS files. This is meant to allow users to reliably detect changes in a file's content.

hpsssum has support for every checksum algorithm that the HPSS checksum library supports. hpsssum also supports storing generated checksums into HPSS metadata using the HPSS File Hashing API interface or the User-defined Attributes (UDA) feature. The hpsssum tool can also read stored checksums from HPSS File Hashing or UDA metadata and verify it against generated checksums. When using the UDA feature, these checksums can be created to be compatible with HPSS-style attribute names, or compatible with legacy hpsssum-style attribute names.



Support in this tool for reading/writing hpsssum-style attributes has been deprecated and will be removed in a future version of HPSS.

The default output mimics the output of similar tools such as md5sum, sha1sum, and others. Given the large file sizes that can be involved, the tool also provides read activity markers as the file is being read. This activity marker can be silenced.

hpsssum can be used in conjunction with other checksum applications, such as VFS with checksum, however hpsssum provides no attribute locking. All attribute updates are transactional, but sets of updates are not guaranteed to be atomic. In general, files still being written should not have their checksum values updated by this tool. Any updates to the file after the checksum has been updated should either be made via an interface that supports updating the checksum attributes, or should be followed by an update of the checksum by this tool.

F.60.4. Options

-c *hash-type*

Type of checksum to generate. Available checksums are sha1, sha224, sha256, sha384, sha512, md5, crc32, adler32.

-f *file-name*

Name of file to checksum

-F *file-list-file*

Name of file containing line-delimited file names

-i *subsystem-id*

Subsystem to search (error and invalid subcommands only)

-R

Use hpss_Read instead of PIO

-S

Allow the mover to select protocol for PIO

-n *num-threads*

Number of threads to use to checksum files

- Each thread checksums one file at a time

-b *buffer size*

Buffer size to read

-L

Local file

-s

Silence activity markers

-a *auth-type*

Used to specify the authentication type to use. Supported values include: unix (Unix authentication), krb5 (Kerberos)

-l

Used to create a new login credential - only valid for *unix* authentication

-p

Used to specify the user principal name

-k

Used to specify login using a keytab file

-r

Used to specify the maximum number of objects by a search request (Invalid or Error)

-t *keytab*

Used to specify the keytab file path

-v

Show verbose output

-h

Show help output

-d

Used to convert timestamps to formatted time strings when printing attributes

-g

Use legacy hpsssum-style attribute names rather than hpss-style attribute names (hpss attribute names are compatible with other checksum interfaces such as VFS). Support for legacy hpsssum-style attribute names will be removed in a future version of HPSS.

-N

Do not stage HPSS files

-u *command*

Used to specify the command type for HPSS File Hashing based checksumming

Command	Description
update	Store Checksum Info
verify	Verify Stored Checksum Info
list	Print Stored Checksum Info
remove	Remove Stored Checksum Info

-U *command*

Used to specify the command type for UDA-based checksumming

Command	Description
verify	Verify Stored UDA Checksum Info
list	Print Stored UDA Checksum Info
remove	Remove Stored UDA Checksum Info
error	List files whose checksum state is *error*
invalid	List files whose checksum state is *invalid*

NOTE: The -f option accepts "-" and will read from stdin. The -f can be supplied multiple times for multiple files. The -F can be supplied multiple times for multiple list files. The -f and -F can both be used, however the total number of files cannot exceed 1000 files. The -c option may be omitted if verifying via UDA; the type will automatically be determined.

NOTE: The error and invalid subcommands do not require a file name or an algorithm, but do require that the user be specified as a *trusted* HPSS user.

NOTE: The -g option should be used when the site has previously used legacy hpsssum with the HPSSSUM checksum keys. Usage of keys in this format by hpsssum are deprecated; the ability to read keys in this format will be removed in a future version of HPSS.

NOTE: For very small files the -R option may improve performance due to the additional overhead in setting up a PIO transfer.

NOTE: SAN3P is off by default; in order to transfer with SAN3P, use the -S option.

F.60.5. Examples

CHECKSUM OF HPSS FILE

To generate a MD5 checksum for an HPSS file, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm
..........
39c6d136ee86fdd57a5f20278800e996  /tmp/testfile
```

LOCAL FILE CHECKSUM

You can also checksum local files using the hpsssum tool. This will be useful for verifying the hpsssum tool because you can run hpsssum and some other checksum tool (like md5sum, sha1sum, etc.) and compare their outputs. The HPSS File Hashing and UDA features are not available when checksumming local files.

```
# /opt/hpss/bin/hpsssum -cadler32 -f /opt/hpss/bin/hpss_core -L -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm
......
cad7c9ec  /opt/hpss/bin/hpss_core
```

GENERATE CHECKSUM WITH HPSS STORAGE

To generate and store a checksum for an HPSS file, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -u update
..........
Update success.

39C6D136EE86FDD57A5F20278800E996  /tmp/testfile
```

LIST CHECKSUM WITH HPSS STORAGE

To list the checksum for an HPSS file, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -u list

File            : /tmp/testfile
Algorithm       : md5
Flags           : <NONE>
```

```
Digest          : 39C6D136EE86FDD57A5F20278800E996
```

VERIFY CHECKSUM WITH HPSS STORAGE

To generate and verify a checksum for an HPSS file, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -u verify
..........
Verify Success /tmp/testfile: stored hash matches current hash.

39C6D136EE86FDD57A5F20278800E996  /tmp/testfile
```

VERIFY OF HPSS FILE (using UDA checksum storage)

To read the data and verify the checksum stored in UDA vs. the current data, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -U verify
..........
39c6d136ee86fdd57a5f20278800e996  /tmp/testfile

Verify Success: stored hash matches current hash.
```

You can also omit the -c option if you are doing verifying a UDA because the tool will just get the checksum type from the UDA:

```
# /opt/hpss/bin/hpsssum -f /tmp/testfile -a unix -k -t /var/hpss/etc/hpss.unix.keytab
-p hpssssm -U verify
Using hash type: md5
..........
39c6d136ee86fdd57a5f20278800e996  /tmp/testfile

Verify Success: stored hash matches current hash.
```

VERIFY OF HPSS FILE (using hpsssum-style UDAs)

To read the data and verify the checksum stored in hpsssum-style UDAs vs. the current data, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -U verify -g
..........
39c6d136ee86fdd57a5f20278800e996  /tmp/testfile

Verify Success: stored hash matches current hash.
```

REMOVAL OF UDA ATTRIBUTES

To remove attributes may also be removed using this tool:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -U remove
Attributes removed.
```

This may leave behind a "stub" attribute named /hpss/hpsssum or /hpss/fs. This is normal behavior.

DETERMINE FILES WITH ERROR STATE (HPSS-style only)

To find files whose state is "error", due to an inability by an application to retrieve the proper information to get or set the checksum, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -U error
Failed to get objects using query: $DOC/hpss/user/cksum[state="Error"], rc=-2
```

The result (-2) indicates that no entries were found. If a file in the error state is found, the result may look like:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -U error
0) ./tmp/testfile
```



If this query is going to be run regularly, the XPath /hpss/user/cksum/state should be added as an XML index using the xmladdindex tool. The type of this index will be *varchar hashed*. On systems which have a large number of files with UDAs, this operation could take a significant amount of time.

DETERMINE FILES WITH INVALID STATE (HPSS-style only)

To find files whose state is "invalid", due to an actual discrepancy by an application to validate the stored with the computed checksum, run:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -U invalid
No objects match the query on the provided subsystem.
```

If a file in the error state is found, the result may look like:

```
# /opt/hpss/bin/hpsssum -c md5 -f /tmp/testfile -a unix -k -t
/var/hpss/etc/hpss.unix.keytab -p hpssssm -U invalid
0) ./tmp/testfile
```



If this query is going to be run regularly, the XPath /hpss/user/cksum/state should be added as an XML index using the xmladdindex tool. The type of this index will be *varchar hashed*. On systems which have a large number of files with UDAs, this operation could take a significant amount of time.

CHECKSUM MULTIPLE FILES (Command line)

You can specify multiple files as well. Using multiple files within one execution of hpsssum will improve performance vs. using a single invocation per file due to the overhead of connecting and authenticating with HPSS. hpsssum enforces a limit of 1000 files per invocation. hpsssum will checksum the files provided in parallel, with each thread working on a single file. The default number of threads is two. The -s flag (silent) may be desirable since all threads will spit out activity markers as they read. The file checksums are guaranteed to be output in the same order as they were input.

```
# ../../bin/hpsssum -a unix -k -t /var/hpss/etc/hpss.unix.keytab -p hpssssm -c md5 -f
/tmp/testfile -f /tmp/testsmall -s
541e36aceaaddc6507554ff2a3a9b6f4  /tmp/testfile
4409e8b51f18bf4c4bd03bc4b1e6980a  /tmp/testsmall
```

CHECKSUM MULTIPLE FILES (File List)

For large groups or groups which should be more persistent than the command line, hpsssum supports reading a file containing line-delimited file names. This execution mode can also be mixed with files on the command line. Again, the file checksum output is guaranteed to be in the same order as the input.

```
# ../../bin/hpsssum -a unix -k -t /var/hpss/etc/hpssx.keytab -p hpssssm -c md5 -F
./file-list -s
541e36aceaaddc6507554ff2a3a9b6f4  /tmp/testfile
4409e8b51f18bf4c4bd03bc4b1e6980a  /tmp/testsmall
```

F.60.6. See Also

md5sum, sha1sum, sha224sum, sha384sum, sha512sum, cksum

F.61. hpsstouch(7)

F.61.1. Name

hpsstouch - Touch an HPSS file

F.61.2. Syntax

hpsstouch *FILE...*

F.61.3. Description

This program *touches* an HPSS file. It opens and immediately closes the file. This may cause a stage request to occur depending upon the COS settings.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

F.61.4. Options

-h, --help
Display usage information.

--authtype *type*
Authentication type

--keytab
Use keytab for auth

--ktpath *keytab-path*
Set keytab path

--newcred
Create new creds (implies password)

--principal *principal*
Use principal name

F.61.5. Example

Example 1: Touch an HPSS file

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpsstouch /tmp/path/to/file
```

F.62. hpss_san3p_part(7)

F.62.1. Name

hpss_san3p_part - Write or display partition table for SAN3P disks.

F.62.2. Syntax

hpss_san3p_part [-f] { -i *device* [*block size*] | -s *device* | -p *device* }
[-?] [-h]

F.62.3. Description

hpss_san3p_part is a utility for partition tables on SAN3P disks.

hpss_san3p_part can be used to create a partition table on a SAN3P disk. This must be done before configuring a SAN3P disk device in HPSS. Creating a partition table on a SAN3P disk will assign a UUID to the disk. This uuid should be entered in the Device Name field when configuring the disk device, instead of the usual device pathname (e.g., /dev/sda).

hpss_san3p_part can also be used to display existing partition table information on a SAN3P disk.

This tool must be run as root.

F.62.4. Parameters

-f

Force overwrite of existing partition table.

-i

Initializes a new partition table, using the specified media block size. The default media block size is 4KB.

-s

Prints summary information for the disk.

-p

Prints the partitions on the specified disk.

-?

Displays the usage message.

-h

Displays the usage message.

device

Device pathname of SAN3P disk. e.g. /dev/sda

block size

Media block size for SAN3P disk. e.g. 8KB Default value is 4KB

F.62.5. Examples

```
$ hpss_san3p_part -f -i /dev/sda 8KB
$ hpss_san3p_part -s /dev/sda
$ hpss_san3p_part -p /dev/sda
```

F.63. hpss_unix_auth(7)

F.63.1. Name

hpss_unix_auth - diagnostic tool to check the HPSS authentication system

F.63.2. Syntax

hpss_unix_auth [-c] [-P] [-U] [-S] [-s *service*] [-k *keytab*]
[-p *password*] [-H] *login*

F.63.3. Description

The hpss_unix_auth utility allows one to verify that the HPSS authentication is working as intended. It can also be used to verify that individual user accounts are up to date with respect to keytabs or passwords.

F.63.4. Options

-c

Cache the temporary credentials on success.

-k *keytab*

File name of a keytab which contains the user's password key. This key will be used in an attempt to authenticate the user. Note that this cannot be used with PAM.

-p *password*

Provide the password on the command line. This is strongly condemned from a security perspective, as other users on the system might see this command during a process list.

-P

Use PAM directly to attempt authentication. This bypasses the HPSS UNIX auth service. It cannot be used with keytabs, and requires root privilege.

-S

Attempt to open a full login session. This simulates a user fully logging into a service such as PFTP, rather than merely verifying that the user's security information is correct. This may be useful for determining why a user cannot log in even if the password is correct. This is only relevant for PAM authentication.

-s *service*

The name of the PAM service which will be used for logging in. The default is "hpss". Note that this does not check that the service is used by any HPSS components, so verifying that a user can log into service xxx may not necessarily provide useful diagnostics where HPSS is concerned. This is only relevant for PAM authentication.

-U

Use the UNIX socket to contact the HPSS UNIX auth service. This is only possible on the same

system that is running the service.

-t *service*

Request a session ticket for a named service account from the HPSS UNIX auth service.

login

Required parameter. This is the name of the user that is to be authenticated.

F.63.5. Examples

Verify that user johndoe can log in

```
$ hpss_unix_auth johndoe
Password:
Authentication succeeded.
```

Verify that user johndoe's keytab is up-to-date

```
$ hpss_unix_auth -k ~johndoe/unix_keytab johndoe
Authentication succeeded.
```

Verify that user johndoe has the correct password at the PAM level

```
$ hpss_unix_auth -P johndoe
Password:
Authentication succeeded.
```

Verify that user johndoe can get an SSM ticket

```
$ hpss_unix_auth -k ~johndoe/unix_keytab -t hpssssm@EXAMPLE.COM johndoe
User: johndoe@EXAMPLE.COM.  Expires: Thu Sep 05 09:45:33 2024
Authentication succeeded.
```

Get a ticket and test session opening

```
$ bin/hpss_unix_auth -c -S dkay
Password:
Authentication successful!
Authentication session end successful!
$ bin/hpss_unix_ticket
997580439:  Cred dkay@EXAMPLE.COM
```

F.64. hpss_unix_group(7)

F.64.1. Name

hpss_unix_group - manage unix groups when using local password and group files

F.64.2. Syntax

hpss_unix_group add [-r *realm*] [-g *gid*] -u *user_list group* [-n]

hpss_unix_group del [-r *realm*] *group* [-n]

hpss_unix_group addmem [-r *realm*] {-u *user_list* [-n] *group* }

hpss_unix_group delmem [-r *realm*] -u *user_list group* [-n]

F.64.3. Description

The hpss_unix_group utility is used to modify information in the UNIX group file when HPSS is configured to use local unix password and group files.

F.64.4. Options

add

The "add" command is used to create a new group and add members to the group.

-g *gid*

The value provided is used as the group id of the group to be added. If not provided, the group file is scanned to find the highest group id. One is added to this value and this becomes the group id.

-u *user list*

List of user names separated by commas. The users in the list are added as members to the group.

group

Required parameter. Name of the group to add.

del

The "del" command is used to delete a group.

group

Name of the group to delete.

addmem

The "addmem" command is used to add members to a group.

-u *user list*

List of user names separated by commas. The users in the list are added as members to the group.

group

Name of the group to add members to.

delmem

The "delmem" command is used to delete members from a group.

-u *user list*

List of user names separated by commas. The users in the list are deleted from the group.

group

Name of the group to delete members from.

Option for all commands

-n

Do not create backup group file

F.64.5. Examples

Create a group named hpss. Set the group id to 1000. Put members test1 and test2 in group.

```
$ hpss_unix_group add  -g 1000  -u "test1,test2"  hpss
```

Delete the group named hpss

```
$ hpss_unix_group del hpss
```

Add members test3 and test4 to the group hpss

```
$ hpss_unix_group addmem  -u "test3,test4"  hpss
```

F.65. hpss_unix_keytab(7)

F.65.1. Name

hpss_unix_keytab - add, delete, update or list unix keytab file entries

F.65.2. Syntax

hpss_unix_keytab [-f *keytab file*] [-o *password* | -O] [-p *password* | -r | -P] add [-U | -S] *principal*

hpss_unix_keytab [-f *keytab file*] [-o *password* | -O] [-p *password* | -r | -P] [-a] update [-U | -S] *principal*

hpss_unix_keytab [-f *keytab file*] del *principal*

hpss_unix_keytab [-f *keytab file*] list

F.65.3. Description

The hpss_unix_keytab utility is used to add, delete, update, or list unix keytab entries for the specified keytab file.

If using HPSS system passwords, keytabs are independent from account passwords. The keytabs themselves are still constructed from passwords, but changing a keytab will not change the account password.

F.65.4. Options

The "add" command is used to create a new keytab entry for the identity specified by <principal>. It is an error if the entry is already present. It uses the following options:

-S

Update the system files directly instead of using the HPSS login service to update the keytab. This bypasses the need for old credentials, but requires that the user run as root on the same host on which the login service runs.

-U

Use the UNIX socket to connect to the HPSS login service

-f *keytab file*

The path name for the keytab file to use. If this option is not provided, then the path specified by the HPSS_UNIX_KEYTAB_FILE environment setting is used. This path typically defaults to /var/hpss/etc/hpss.unix.keytab.

-p *password*

The value provided is encrypted and stored as the principal's new password in the keytab entry. If not using system files, a current credential (keytab or current password) must be provided to authenticate the password change.

-P

The user is prompted for the new password value. If not using system files, a current credential (keytab or current password) must be provided to authenticate the password change. If a current keytab is not already in the keytab file, use -o or -O to provide the current UNIX account password.

-o *password*

The value provided authenticates the user's current password to the HPSS login service. If neither -o or -O are provided, and if not using the system files directly, the keytab will be used for authentication.

-O

Prompt for the user's current password.

-r

Indicates that a new random password is to be generated, encrypted and stored as the principal's password in the security database (i.e. the local HPSS password) and the keytab entry. If not using -S, a current credential must be valid (keytab entry or provided current password).

principal

Required parameter. This is the name of the principal to be used for adding the new keytab entry.



The -P, -p, and -r flags are mutually exclusive and are only valid when using an HPSS password database. Likewise, -o and -O are mutually exclusive.

The "update" command is used to update the keytab password entry for the identity specified by <principal>. It has almost all of the same semantics as the "add" command except that it is normally an error if the desired <principal> does not already exist. It adds the following option:

-a

Add the <principal> to the keytab if it does not already exist.

The "del" command is used to delete the keytab entry for the identity specified by <principal>. It uses the following options:

-f *keytab file*

The path name for the keytab file to use. If this option is not provided, then the path specified by the HPSS_UNIX_KEYTAB_FILE environment setting is used. This path typically defaults to /var/hpss/etc/hpss.unix.keytab.

principal

Required parameter. This is the name of the principal to be used for deleting the keytab entry.

The "list" command is used to list the keytab entries. It uses the following option:

-f *keytab file* The path name for the keytab file to use. If this option is not provided, then the path specified by the HPSS_UNIX_KEYTAB_FILE environment setting is used. This path typically defaults

to /var/hpss/etc/hpss.unix.keytab.

F.65.5. Examples

Create a keytab entry for principal testuser1. Set the password to test. The old password was "old" and the keytab did not have a current entry.

```
$ hpss_unix_keytab -o "old" -p "test" add testuser1
```

Update the keytab entry for principal testuser1. Randomize the password.

```
$ hpss_unix_keytab -r update testuser1
```

Delete the keytab entry for principal testuser1 in the /tmp/keytab.unix file.

```
$ hpss_unix_keytab -f /tmp/keytab.unix del testuser1
```

List all entries in the default unix keytab file

```
$ hpss_unix_keytab list
```

Change the password and update the keytab for principal user1 bypassing the HPSS Login Service

```
$ hpss_unix_keytab update -P -S user1
```

F.65.6. Files

/var/hpss/etc/hpss.unix.keytab

Default unix keytab file

F.65.7. See Also

hpss_unix_user hpss_unix_group hpss_unix_passwd

F.66. hpss_unix_passwd(7)

F.66.1. Name

hpss_unix_passwd - change passwords for UNIX users when using a local password file

F.66.2. Syntax

hpss_unix_passwd [-d] [-n] [-p] [-P | -S | -U] [-r *realm*] *login*

F.66.3. Description

The hpss_unix_passwd utility allows a user or an administrator with proper authority to change the passwd of a specific user.

When the command is issued, the caller will be prompted for the old password unless the caller is root. If the caller is prompted for the password, the old password must be entered twice correctly and then the caller is prompted for the new password. Once the new password is entered, it is encrypted and stored in the password entry in the password file.

F.66.4. Options

-d

Delete the password. This disables login for the user.

-n

Do not create backup passwd file

-p

Do not prompt for a password. If the old password is required, the attempt will fail. Intended for scripts.

-r *realm*

Specify the realm. If the user is not in the default realm, this flag and option are required.

-P

Use PAM directly to change the password. May require root.

-S

Operate on the shadow file directly. Requires root.

-U

Connect to the HPSS login service with the UNIX socket. Must be on the same host as the login service.

F.67. hpss_unix_user(7)

F.67.1. Name

hpss_unix_user - create and delete unix users using local passwd and group files

F.67.2. Syntax

hpss_unix_user add [-r *realm*]
[-c *comment*]
[-d *home dir*]
[-g *primary group*]
[-G *group*[,...]]
[-n]
[-s *shell*]
[-u *uid*]
[-p *passwd*]
login

hpss_unix_user del *login*
[-n]

F.67.3. Description

The hpss_unix_user utility is used to add or delete users to the HPSS system when using the option of local password and group files.

F.67.4. Options

The "add" command is used to create a new user and uses the following options.

-c *comment*

The string provided is stored in the gecos field of the password entry. This option is not required and the default provided is "HPSS login entry". Typically the full name of the user is provided.

-d *home dir*

String provided is used to set the user's home directory in the password entry. If not provided the default is /home/xxxx where xxxx is the value of the login parameter provided.

-g *primary group*

String provided is used to set the primary group id in the password entry . If no value is provided the default is "hpss".

-G *group*[,...]

A list of group names separated by commas should be provided. The user is added to each one of the groups specified.

-n

Do not create backup user file

-p *passwd*

Required parameter. The value provided is encrypted and stored as the value of the user's password in the password entry.

-r *realm*

Specify the realm for login

-s *shell*

String provided is stored as user's shell in the password entry. If not provided, the default is "/bin/ksh".

-u *uid*

Value provided is stored as the uid associated with the user in the password entry. If not provided, the highest uid in the passwd file is determined and the uid is set to this value plus one.

login

Required parameter. This is the name of the user that is to be added.

The "del" command is used to delete a user and uses the following option

login

Name of the user to delete.

F.67.5. Examples

Create a user name testuser1. Set the passwd to test. Set the user's uid to 1000. Set the user's primary group to nobody. Add the user to the group hpss. Default everyting else.

```
$ hpss_unix_user add  -u 1000  -p "test" -g "nobody" \
  -G "hpss"   testuser1
```

Delete the user name testuser1

```
$ hpss_unix_user del testuser1
```

F.68. hpss_update_priority_by_create_time(7)

F.68.1. Name

hpss_update_priority_by_create_time - Update priority by create time

F.68.2. Synopsis

hpss_update_priority_by_create_time -c *SC* [-v] [-h *HPSS Root*] [-F *family*]

F.68.3. Description

This tool orders the selection of volumes for write I/O based by applying weighting criteria (the volumes' create time).

F.68.4. Required Arguments

-c

Storage class to operate upon. The tool can only operate on one storage class (SC) at a time.

F.68.5. Optional Arguments

-h

HPSS root directory. Default /opt/hpss.

-v

Verbose output.

-F

File family to operate upon. The tool can only operate on one family at a time.

F.68.6. Prerequisites

The following software prerequisites are required:

- Python 3.6 or later.
- the ibm-db Python package (available via *pip install ibm_db* or here: https://pypi.org/project/ibm-db/)

F.69. hpss_update_priority_by_pvr(7)

F.69.1. Name

hpss_update_priority_by_pvr - Update priority by PVR

F.69.2. Synopsis

hpss_update_priority_by_pvr -c *SC* [-h *HPSS root*] [-v] [-f *config*]
[-F *family*]

F.69.3. Description

This tool orders the selection of volumes for write I/O based by applying weighting criteria (the volumes' PVR) as described in the provided configuration file.

F.69.4. Configuration

This tool uses a configuration file to describe the PVRs and the weights. A default configuration file can be generated using hpss_create_default_vv_select_weights.

The PVRs are identified in the "Name" field as "PVR<ID>", where ID is the server ID. The weights are integers greater than 0. The relative weights determine the ratio of the volume selection between the various PVRs that are configured. Any volumes associated with a PVR which does not appear in the weights will be placed at the end of the prioritized list.

Example JSON configuration file:

```
{ "SamePVRPenalty": 2,
  "Weights": [
              { "Name": "PVR7", "Weight": 6 },
              { "Name": "PVR5", "Weight": 4 }
            ]
}
```

The above indicates that PVR 7 has a weight of 6 and PVR 5 has a weight of 4. In a simple, non-striped case, if you had an even number of volumes in each PVR, you would see for every 10 volumes selected, 6 would be from PVR 7 and 4 would be from PVR 5.

In striped cases, the PVR weights for each volume are applied across the stripe cumulatively. For example, a 2-way striped storage class with volumes in PVR7+PVR5 have a weight of 10, while a volume spanning PVR5/PVR5 would have a weight of 8, and a volume of PVR7+PVR7 would have a weight of 12.

The "SamePVRPenalty" is a weighting penalty applied in striped volume cases. The "SamePVRPenalty" applies a one time penalty to volumes which use the same PVR multiple times, in this example reducing the weight of PVR5+PVR5 to 6 and PVR7+PVR7 to 10, while leaving PVR7+PVR5 at 10. The SamePVRPenalty will not cause the weight to become 0 or negative.

F.69.5. Required Arguments

-c

Storage class to operate upon. The tool can only operate on one SC at a time.

F.69.6. Optional Arguments

-h

HPSS root directory. Default /opt/hpss.

-v

Verbose output.

-f

Alternate weighting config file location. Default HPSS_ROOT/tools/ss/vv_select_priority/config_pvr.

-F

File family to operate upon. The tool can only operate on one family at a time.

F.69.7. Prerequisites

The following software prerequisites are required:

- Python 3.6 or later
- the ibm-db Python package (available via *pip install ibm_db* or here: https://pypi.org/project/ibm-db/)

F.70. hpss_update_priority_by_pvr_drive(7)

F.70.1. Name

hpss_update_priority_by_pvr_drive - Update priority by PVR and drive type

F.70.2. Synopsis

hpss_update_priority_by_pvr_drive -c *SC* [-h *HPSS root*] [-v]
[-f *config*] [-F *family*]

F.70.3. Description

This tool orders the selection of volumes for write I/O based by applying weighting criteria (the volumes' PVR and drive type) as described in the provided configuration file.

F.70.4. Configuration

This tool uses a configuration file. A template is provided in /opt/hpss/tools/ss/vv_select_priority/config_pvr_drive. The user can specify the configuration file to adjust the weighting criteria.

The configuration file describes the PVRs, drive combinations, and weights. The PVRs are identified in the "Name" field as "PVR<ID>/<DRIVE>". The PVR ID is the server ID. The drives are identified by their HPSS drive type and subtype. These can be viewed using `hpss_vv_select_priority`.

The weights are integers greater than 0. The relative weights determine the ratio of the volume selection between the various criteria that are configured. Any volumes associated with criteria which does not appear in the weights will be placed at the end of the prioritized list.

Example JSON configuration file:

```
{ "SamePVRPenalty": 2,
  "Weights": [
                { "Name": "PVR7/LTOGEN5", "Weight": 5 },
                { "Name": "PVR7/LTOGEN6", "Weight": 3 },
                { "Name": "PVR11/LTOGEN5", "Weight": 5 }
  ]
}
```

The above indicates that PVR 7 / LTO 5 has a weight of 5, PVR 7 / LTO 6 has a weight of 3, and PVR 11 / LTO 5 has a weight of 5. In a simple, non-striped case, if you had an even number of volumes in each PVR, you would see for every 13 volumes selected, 5 would be from PVR 7 / LTO 5, 3 would be from PVR 7 / LTO 6, and 5 would be from PVR 11 / LTO 5.

In striped cases, the weights for each physical volume are applied across the stripe cumulatively. For example, a 2-way striped storage class with volumes in PVR7/LTOGEN5+PVR11/LTOGEN5 have a weight of 10, while a volume spanning PVR7/LTOGEN6+PVR11/LTOGEN5 would have a weight of 8,

a volume spanning PVR7/LTOGEN5+PVR7/LTOGEN6 would have a weight of 8, a volume spanning PVR11/LTOGEN5+PVR11/LTOGEN5 would have a weight of 10, and so on.

The "SamePVRPenalty" is a weighting penalty applied in striped volume cases. The "SamePVRPenalty" applies a one time penalty to volumes which use the same PVR multiple times, in this example reducing the weight of PVR11/LTOGEN5+PVR11/LTOGEN5 to 8 and PVR7/LTOGEN5+PVR7/LTOGEN6 to 6, while leaving PVR7/LTOGEN5+PVR11/LTOGEN5 at 10. SamePVRPenalty will not cause the weight to become 0 or negative.

F.70.5. Required Arguments

-c

Storage class to operate upon. The tool can only operate on one SC at a time.

F.70.6. Option Arguments

-h

HPSS root directory. Default /opt/hpss.

-v

Verbose output.

-f

Alternate weighting config file location. Default HPSS_ROOT/tools/ss/vv_select_priority/config_pvr_drive.

-F

File family to operate upon. The tool can only operate on one family at a time.

F.70.7. Prerequisites

The following software prerequisites are required for the hpss_update_priority_by_create_time tool:

- Python 3.6 or later
- the ibm-db Python package (available via *pip install ibm_db* or here: https://pypi.org/project/ibm-db/)

F.71. hpssuser(7)

F.71.1. Name

hpssuser - HPSS User Management Utility

F.71.2. Syntax

hpssuser -add *user type(s)* [*options*...]

hpssuser -del *user type(s)* [*options*...]

hpssuser -list *user*|all *type(s)*

hpssuser -ssmclientpkg *path* [*options*]

hpssuser -h

F.71.3. Description

The hpssuser utility is used to add, delete or list HPSS and SSM users, or to create an SSM client package.

Arguments for the hpssuser utility are used to specify actions, user types, and options. A single action is required and is applied to any user types specified. Most options are used to indicate the values needed to take the requested action for the user type(s). If a value is needed which was not provided by an option, then the user will be prompted for the value.

F.71.4. Actions

-add *user*

The add action is used to create a new user entry for the identity specified by <user>.

-del *user*

The delete action is used to remove a user entry for the identity specified by <user>.

-list *user*|all

The list action is used to list the identity specified by <user>. If the keyword "all" is provided, then all users of the requested type are listed. Listing all Unix users is not supported.

-ssmclientpkg *path*

The ssmclientpkg action is used to create a tar file containing the files needed to run the HPSS graphical user interface.

-h | -help

The help option shows a brief help message.

F.71.5. User Types

-acct

Indicates that the action specified is for a user account. The type will be determined through parsing the site.conf file.

-ssm

Indicates that the action specified is for an SSM user.

-all

Indicates that the action specified is for all valid user types for the HPSS configuration. This will always include the -acct and -ssm types.

F.71.6. Options

These options are used with the add action and apply to one or more user types

-default unix

Use the specified type of existing data as default values. This option is used by all user types.

-fullname *name*

The descriptive user name. This option is used by the unix and ftp user types. If the <name> string contains spaces, it must be enclosed in quotes.

-gecos *text*

Set the HPSS account ID. This option applies to the ldap user type.

-gid *id*

The numeric primary group ID. This option is used by the ldap user type, and the ftp user type when PFTP is configured to use its own password file.

-group *name*

The primary group name. This option is used by the unix user type.

-home *path*

The path of the Unix home directory. This option is used by the unix user type, and the ftp user type when PFTP is configured to use the Unix password file.

-hpsshome *path*

The HPSS home directory. This option is used by the ldap user type, and the ftp user type when PFTP is configured to use its own password file.

-nohome

When adding an FTP user, don't create a home directory in HPSS. Creating a home directory requires that the core server is running.

-password *password*

The user's password. This option is used by the unix, ftp and krb user types.

-shell *path*

The path to the user's login shell. This option is used by the ftp and unix user types.

-uid *id*

The numeric user ID. This option is used by the ftp, ldap and unix user types.

-uuid *uuid*

The UUID for the user. This option applies to the ldap action.

These options apply to either the add or del action

-keytab *path*

When adding or deleting an account, add or delete the user's entry from the keytab specified by <path>.

-nobackup

Do not create backup user, group, or passwd files

This option applies to the ssm action

-ssmlevel *level*

Set the SSM privilege level. Valid choices are admin or operator. Refer to the HPSS Management Guide for a description of security levels.

This option applies to the ssmclientpkg action

-packagedoc

Include HPSS documentation in the SSM client package.

F.71.7. Example

The following command will create a user and give the user SSM privileges, prompting for needed input.

```
# hpssuser -add testuser -acct -ssm
```

The following command will create a user account. All required options are included so the user will not be prompted for any input.

```
# hpssuser -add testuser -acct -password mypassword -uid 900 -group hpss \
  -fullname  "Test User" -shell /bin/bash   -home /home/testuser
```

The following command will create an SSM client package with the HPSS documentation included.

```
# hpssuser -ssmclientpkg /tmp/ssmpkg.tar -packagedoc
```

F.71.8. Notes

The utility must be run on a machine which has access to the relevant Unix, LDAP and Kerberos data sources.

When creating HPSS home directories the core server must be running. hpssuser will attempt to use the *root* principal and the /var/hpss/etc/root.keytab authenticator.

If the system is configured to use Unix authentication and hpssuser is running as root, then hpssuser will attempt to use the *root* Unix identity and the /var/hpss/etc/root.unix.keytab authenticator. Otherwise, the user will be prompted for a principal name and keytab.

The principal and authenticator used to authenticate to HPSS when creating the user's home directory may also be specified by setting the HPSSUSER_ADM and HPSSUSER_ADM_KEYTAB environment variables. This is useful if the *root* user has not been granted administrative authority in the global HPSS configuration, and a different adminstrative account or authenticator should be used.

When the realm is configured for Kerberos operations, unless hpssuser is run as root on the KDC, it must authenticate as a kerberos administrative principal. The environment variables KADMIN_PRINC, KADMIN_KEYTAB and KADMIN_PASSWORD can be used to specify the kerberos principal, and keytab or password to use for kerberos operations. If these environment variables are not set, and hpssuser is executed as root, then it will attempt to authenticate using /var/hpss/etc/root.keytab. Otherwise it will prompt for a principal name and password.

When generating a keytab file, the utility will attempt to set ownership of the keytab to the identity provided with the add action. If this user does not have a system Unix account on the system, ownership of the keytab file will be set to root and a warning message will be displayed.

F.71.9. Environment

HPSSUSER_ADM

The name of the administrative account used to create a user's HPSS home directory.

HPSSUSER_ADM_KEYTAB

The name of the keytab file which contains an entry for the HPSSUSER_ADM.

KADMIN_PRINC

The principal to use for kerberos administrative operations.

KADMIN_KEYTAB

The keytab to use for kerberos administrative operations.

KADMIN_PASSWORD

The password to use for kerberos administrative operations. If both KADMIN_KEYTAB and KADMIN_PASSWORD are specified, KADMIN_KEYTAB will be used.

F.71.10. Files

/var/hpss/etc/site.conf
HPSS site and realm file

/var/hpss/etc/env.conf
HPSS environment file

/var/hpss/etc/group
HPSS local group files

/var/hpss/etc/passwd
HPSS password file

/var/hpss/etc/HPSS.conf
HPSS Configuration file

/var/hpss/ftp/etc/ftppasswd
HPSS FTP password File

F.71.11. See Also

hpss_ldap_admin, hpss_unix_group, hpss_unix_passwd, hpss_unix_keytab, hpss_unix_user, scrub

F.72. hpss_vv_select_priority(7)

F.72.1. Name

hpss_vv_select_priority - VV Select Priority

F.72.2. Synopsis

hpss_vv_select_priority [--help] [--reset | --input-file *INPUT_FILE* | --list]
--sc *SC* [--family *FAMILY*]
[--subsys-db-name *SUBSYS_DB_NAME*]
[--limit *LIMIT*] [--schema *SCHEMA*]

F.72.3. Description

hpss_vv_select_priority allows an administrator to list, reset, or update VV selection priority for a given storage class (SC).

This tool manages the priority associated with tape VVs for their selection in HPSS. This selection only applies to selecting new volumes for use in writing; it does not apply to how tapes already written will be selected for future write operations, or how the order in which tape volumes with pending read requests will be read.

The tool has three modes of operation:

- A list mode, where it displays information associated with a set of VVs in a given storage class.
- A reset mode, where each volume in the storage class is assigned the same, large priority.
- An update mode, where an input file containing VVIDs and priorities is ingested and used to update the specified volumes.

This tool is meant to be used as a building block for building more complex prioritization workflows, either using HPSS-provided tools or site-supplied tools. However it can also be used to do special purpose, one-off prioritization of volumes.

The priorities are selected based on the lowest value first - that is, the VV at priority 1 will be selected before a VV at priority 2.

F.72.4. Required Arguments

--sc SC

Storage class to operate upon. The tool can only operate on one SC at a time.

--reset | --input-file INPUT_FILE | --list

One of these actions must be provided. The --input-file action requires an input file be specified. The input file format is in a line-delimited form of VVID,PRIORITY. For example:

0000053904000000020 1ee1b9deb563f760000,1

Note that the tool does not modify any volumes not provided in the input file. It is perfectly acceptable for multiple volumes to have the same priority value - they are then ordered by creation time. For example, another volume may already have a priority of 1. In order to ensure that all volumes have a unique priority, any existing volumes with that priority must be explicitly updated to some other priority. One simple way to do that is to periodically apply selection criteria to all volumes or a large subset of volumes at once.

F.72.5. Optional Arguments

-h, --help

show a help message and exit

--subsys-db-name SUBSYS_DB_NAME

the subsystem database name (default: *hsubsys1*)

--schema SCHEMA

database schema name (default: *hpss*)

--cfg-db-name CFG_DB_NAME

config database name (default: *hcfg*)

--limit LIMIT

the number of VVs to process when listing volumes

F.72.6. Prerequisites

The following software prerequisites are required:

- Python 3.6 or later
- the ibm-db Python package (available via *pip install ibm_db* or here: https://pypi.org/project/ibm-db/)

F.73. hpssxfind(7)

F.73.1. Name

hpssxfind - Search user-defined attributes

F.73.2. Syntax

hpssxfind [-i *subsys*] [-h] [-p] [-x *find xquery*]
[-t *numthreads*] [-j] [-q] *xquery*

F.73.3. Description

This program searches HPSS user-defined attributes and returns matching objects. There are two modes of operation for the tool, object-only mode and xquery find mode.

Object Only Mode

Object Only Mode is the default behavior of the tool. All objects matching the input XQuery are returned. By default, the tool will resolve the objects into HPSS full paths. If the path information is not required, the -q option will skip path resolution and simply dump the matching object identifiers. Path resolution is often a very expensive part of the search process.

XQuery Find Mode

XQuery Find Mode takes a secondary xquery. In database parlance, this is the portion of the xquery information to return - NOT to match against. Whereas, the primary xquery is the "where" criteria that indicates which objects to select. These may be non-overlapping queries. Like Object Only Mode, XQuery Find Mode will by default perform path resolution on the results.

HPSS user authentication uses the current system user, or the **HPSS_CLI_USER** environment variable to specify the HPSS principal. The authentication mechanism can be provided via the **HPSS_PRIMARY_AUTHN_MECH** environment variable (krb5 or unix).

Keytab location can be specified using **HPSS_UNIX_KEYTAB_FILE** for UNIX authentication, and **HPSS_KRB5_KEYTAB_FILE** for Kerberos authentication.

All of the above options may also be specified on the command line.

Since some short flags are used for multiple purposes, those necessary for authentication must be listed first while those that control the copy are listed second. The first flag that is unique to copying will make the rest of the flags be understood as copy options. One may also force following options to be copy options by using the — flag before the first copy option. The full-length flags will always be unambiguous, but the auth flags must still be in the first group.

F.73.4. Options

-h, --help

Display usage information.

--authtype *type*

Authentication type

--keytab

Use keytab for auth

--ktpath *keytab-path*

Set keytab path

--newcred

Create new creds (implies password)

--principal *principal*

Use principal name

-i *subsys*

Which subsystem to search. Only one subsystem can be searched at a time. The default subsystem is 1.

-p

Perform path name resolution. This provides path information for matching objects. By default this is on.

-x *find xquery*

Display matching path names and XML values specified by find xquery. See the description of XQuery Find Mode above.

-j

Display UDA values as JSON.

-q

Quick mode. This disables path name resolution, so only object ids and results are provided. Path resolution is an expensive operation.

-t *thread count*

Thread count. This sets the number of threads to use for path resolution. This ideally will be similar to the client system core count. The default is 8.

F.73.5. Example

See the help output from -h for more examples. See the Db2 XPath reference for more information on xquery and xpath. XPath supports and/or and conditionals.

Example 1: Find HPSS files with GHI attributes and a migration index of 4

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
```

```
# hpssxfind '$DOC/hpss/ghi[Midx=4]'
```

Example 2: Find HPSS files with an application-set expiration date UDA that has already passed

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssxfind '$DOC/hpss/myapp[expiration < 1624982899]'
```

Example 3: Retrieve all application-set information when a time has passed

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssxfind -x '$DOC/hpss/myapp  '$DOC/hpss/myapp[finalized < 1624982899]'
```

Example 4: Find HPSS files with GHI attributes and a migration index of 4 without path resolution

```
# export HPSS_CLI_USER=myuser
# export HPSS_UNIX_KEYTAB_FILE=/path/to/keytab
# hpssxfind -q '$DOC/hpss/ghi[Midx=4]'
```

F.74. lbp_verify(7)

F.74.1. Name

lbp_verify - Verify tape volumes via logical block protection

F.74.2. Syntax

lbp_verify [*OPTIONS*] --volume=*vv*

lbp_verify [*OPTIONS*] --restart-at-storage-segment=*ssid*

lbp_verify [*OPTIONS*] --storage-class=*sc*

lbp_verify [*OPTIONS*] --file-list=*fl*

lbp_verify [*OPTIONS*] --delete-checkpoint=*dc*

F.74.3. Description

This program obtains a list of virtual volumes managed by the Core Server and issues a verify request against volumes in that list. That verify request will read the active segments on the tape and compare the block data with the computed checksums stored on the tape. The verify request will also validate that the number of blocks verified is not less than expected, and other cross-checking where possible. If errors are found, they are reported back to the Core Server. When the virtual volume verification is complete, the tool will add a PVHistory entry for each PV. This can be done serially or in parallel, up to a certain number of tape drives. Result data is presented after each volume completes verification.

Values for the subsystem ID, global database name, schema name, authentication mechanism, authenticator and principal name will be taken from the user's environment if the corresponding arguments are not supplied on the command line. If these environment variables do not appear in the user's environment, default values will be used. The command line arguments override the environment.

The user must specify an input source for the volume list, either --volume, --file-list, or --storage -class.

F.74.4. Options

--authentication-mechanism=*am*

-m *am*

The security mechanism used to authenticate the client. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--authenticator-name=*an*

-k *an*

The authenticator which is used to obtain credentials. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--db-schema-name=*sn*

-s *sn*

The ASCII name of the Db2 schema that contains the global database. If not specified on the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--global-db-name=*dn*

-g *dn*

The ASCII name of the global database where the basic configuration information for HPSS is found. If not specified on the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--login-principal=*lp*

-p *lp*

The server principal name under which this program will run. Dump_sspvs is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided on the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--subsystem-id=*ss*

-i *ss*

The number of the subsystem that contains the Core Server of interest. This argument must be a positive non-zero value. If not provided on the command line, the value of the HPSS_SUBSYS_ID environment variable will be used.

--volume *vn*

-v *vn*

Specify a single volume to process.

--file-list *fl*

-f *fl*

Specify a file containing a list of volumes to process.

The file will contain a single volume name per line, e.g. A00001 A00002 etc.

--storage-class *sc*

-c *sc*

Specify a single storage class to process.

--restart-at-storage-segment *storage-segment-id*

-r *storage-segment-id*

Specify the storage segment id at which to begin verification for a single volume.

For example:

```
00000539-07-00000003-01e695843e182bf4-1d2f
or
```

```
x'00000539070000000301e695843e182bf41d2f'
```

--file-family *fi*

-F *fi*

Restrict processing to volumes in the provided file family. Note that this restricts the volumes processed via some input specification method (e.g. storage class, file list)

--time-since-verification=*ts*

-T *ts*

Only verify volumes which have not been verified since the provided time (in days since current time).

--down-volume

-D

Causes HPSS to only allow reads from the verification tool on this volume. Any other reads attempted against the volume while a verification request is ongoing will fail. This can be used in dual copy hierarchies where auto-retry is enabled to continue to make data available while the primary copy is being verified.



This option should only be use when the set of volumes being verified has a valid second copy unless read failures from other applications is acceptable.

--eom-only

-E

Only process EOM volumes.

--skip-retired

-R

Skip processing retired tape volumes.

--drives-to-use

-d *md*

By default, if this option is not specified, a single volume at a time is processed (leading the maximum number of drives to use as variable based upon the inputs). If this is specified, multiple volumes can be processed, up to the drive count specified. Any volumes whose stripe width exceeds the maximum drive count are skipped.

The processing works by spawning threads to work volumes in the list, until those volumes would occupy the provided drive count, and then waiting until a volume completes processing before attempting to find another volume to work on. The tool does not try to reorder the input list in order to use more drives at once.

--volume-count *vc*

-n *vc*

Restricts the number of volumes to be processed.

--output-file *of*

-o *of*

This option redirects the verification status updates to an output file. Error output continues to go to stdout.

--list-lastverify

-l

Only report the last verification time of each volume in the input list.

--list-checkpoint

-L

Only list active checkpoints for each volume in the input list.

--delete-checkpoint *dc*

-x *dc*

Delete any active checkpoints for the specified volume.

--verbose

-V

Print verbose output.

--help

-h

Print help output.

F.74.5. RAIT Tape Volumes

Due to the need to verify all data in order to declare the volume as being verified, RAIT volumes will not use their normal minimal mount characteristics. RAIT volumes processed by this tool must mount their full data + parity stripe width in order to be verified.

F.74.6. Admin Abort

By default, the HPSS Core Server will group together contiguous tape sections to facilitate verification performance with a large limit (defined by HPSS_CORE_VERIFY_SECTION_LIMIT). The HPSS Mover will then submit a very large verify request to the tape drive, optimized for performance. This operation must complete before the Mover can process any abort requests. To make abort processing more responsive, an admin may change HPSS_CORE_VERIFY_SECTION_LIMIT in env.conf and reinit the Core Server. Consider that often each tape mark is going to require close to Seconds Between Tape Mark (SBTM) to process, so for instance of a storage class has a SBTM of 45 and the limit is set to 10, it may require up roughly 7 minutes to process abort the request. Note that reducing this value has a moderate negative impact on verify performance.

F.74.7. Premature Verify Halt

Multiple device errors during a verify operation can cause the verify to halt prematurely — before

the tape volume has been completely verified. When this occurs, there will be device errors in the HPSS log along with this message:

```
MOVR0301 LBP Verify halted due to recurring errors
```

The HPSS_MOVER_VERIFY_ERROR_THRESHOLD controls how aggressively HPSS will halt lbp_verify instances. By default after five device errors a verify will halt. Increasing the HPSS_MOVER_VERIFY_ERROR_THRESHOLD will allow HPSS to tolerate more device errors before halting the verify operation. To allow a verify operation to continue regardless of how many device errors occur, set HPSS_MOVER_VERIFY_ERROR_THRESHOLD to zero (0). This environment variable must be set in ${HPSS_PATH_ETC}/env.conf on the hosts where the remote tape mover processes (hpss_mvr_tcp) run. It affects all verify operations being executed by tape movers where the environment variable is set.

F.74.8. Examples

Verify the volume containing PV name *000718*

```
$ lbp_verify -v 000718
Verify volume '000718'
Complete: Volume: 00071800 Sections: 1001 Blocks: 207531 Errors: 0
```

Verify every volume in /tmp/vollist, using up to 5 drives

```
$ lbp_verify --subsystem-id=1 -f /tmp/vollist -d 5
Verify volume list '/tmp/vollist'
Complete: Volume: PASS0100 Sections: 2 Blocks: 6 Errors: 4
Complete: Errors on: PASS0100 at
    Section: 1 Blocks: 1 Verify Error: 1 Errno:  1
    Section: 2 Blocks: 5 Verify Error: 1 Errno:  5
Complete: Volume: PASS0200 Sections: 2 Blocks: 6 Errors: 0
Complete: Volume: PASS0400 Sections: 2 Blocks: 6 Errors: 0
Complete: Volume: FAIL0100 Sections: 2 Blocks: 6 Errors: 0
GetVVFromPV: core_GetPVSoid returns status=-2 rpc_error=0
GetVVFromPV returns -2

Volume         Time                           Action         Result
PASS0100       Fri Jan 15 16:13:24 2016       Processed      Verify Fail
 (file hash verification failure)
PASS0200       Fri Jan 15 16:13:24 2016       Processed      Verify Ok
PASS0400       Fri Jan 15 16:13:24 2016       Processed      Verify Ok
FAIL0100       Fri Jan 15 16:13:24 2016       Processed      Verify Ok
FAIL0200       Fri Jan 15 16:13:24 2016       Processed      Verify Fail
 (No such file or directory)
```

Only list last verification time on storage class 1000

```
$ lbp_verify --subsystem-id=1 -c 1000 -l
Report on Storage class '1000'
PV: 00071800, Last Verified: Fri Jan 15 16:11:52 2016
```

Only list checkpoint info for storage class 1000

```
$ lbp_verify --subsystem-id=1 -c 1000 -L
Report on Storage class '1000'
PV: 00071800 at section 500, Checkpoint Updated Fri Jan 15 16:15:35 2016
```

Verify all volumes which have not been verified since the timestamp

```
$ lbp_verify --subsystem-id=1 -c 1000 -T 1
Verify Storage class '3'
Complete: Volume: 00071800 Sections: 501 Blocks: 102376 Errors: 0
Complete: Volume: 00071900 Sections: 1501 Blocks: 207331 Errors: 0

Volume         Time                           Action          Result
00071800       Fri Jan 15 16:19:28 2016       Processed       Verify Ok
00071900       Fri Jan 15 16:29:28 2016       Processed       Verify Ok
```

F.74.9. See Also

hpsssum, dump_sspvs, lspvhist, lsvol

F.75. lori(7)

F.75.1. Name

lori - HPSS smart read tool

F.75.2. Syntax

lori [*OPTION*]... [*INPUT*]...

F.75.3. Description

Lori is a utility which utilizes the HPSS batch read functions to provide an out-of-the-box mass read capability for HPSS. It is meant to allow users to read many files from a variety of input sources in an organized fashion. Users of Quaid will recognize Lori as an equivalent tool for reading directly out of HPSS.

Lori is engaged in four steps: 1. Scanning inputs from the sources 2. Applying filters to only stage matching files 3. Coalescing matching files from a minimal number of VVs into a set of batch stage requests 4. Issuing file read requests and providing status updates on the outcome of each file read

Based upon user inputs, a set of input sources is defined. If these are file lists, directories, or UDA queries, then the intermediate results are placed as temporary files to allow for an arbitrary number of items to be processed. These input sources may themselves generate additional input sources to scan and process if recursion is enabled.

From those defined input sources, file objects are identified and their attributes read. At this point, filters specified by the user are applied to allow matches to remain as inputs to be processed, while non-matches are rejected.

Next, the remaining set of files is processed further to build a set of batch stage requests which utilizes a minimal number of tape VVs for each batch. The batches may be processed in parallel based upon user criteria.

Finally, the batches are issued. As reads complete, their status is posted to stdout.

Once all reads have processed, the tool returns to step 1 to retrieve additional inputs from the available data sources and continues on.

The client stripe width can be modified by setting **HPSS_API_PIO_STRIPE_WIDTH** in the environment. The client transfer size can be modified by setting **HPSS_API_PIO_BLOCK_SIZE** in the environment. These values can be used to tune the environment for improved transfer performance.

F.75.4. Input Methods

-f, --file *file-name*

Specify an HPSS file name to read.

-F, --file-list *file-list-file*

Specify a file containing HPSS file names to read.

-B, --bitfile-list *bitfile-list-file*

Specify a file containing HPSS bitfile IDs to read.

Note that this file can contain either bitfile IDs or bitfile ID and distribution key pairs in the form BFID:KEY, for example:

```
00000539-07-00000003-01e93ad4493e9a66-1a39
00000539-07-00000003-01e93ad4493e9a66-1a39:1231
```

-u, --xquery-match *file-match-xquery*

Specify an XQuery, run against the HPSS namespace, whose matches will be read. For example, -u *$DOC/hpss/test* will match any files with an UDA path of /hpss/test.



Any XQuery paths queried should be indexed! Talk with your HPSS administrator to set up an index.

F.75.5. Options

-a, --auth-type *auth-type*

Authentication type.

-p, --login-principal *user*

User Principal.

-k, --keytab

Indicates a keytab will be used.

-t, --keytab-file *keytab*

Keytab path.

-b, --files-per-batch *batch-size*

Number of files to be processed within a batch (default=2000).

-d, --data-buffer-size *buffer-size*

Size of the buffer to use for copying a file with parallel I/O (default=16 MiB)

-D, --stage

Stage data on open prior to being read, if the COS behavior allows it. Lori is a tool for reading data, so there is no need to stage the data unless there will be a need to re-read the data again later. Staging the data prior to reading could reduce overall performance. (default=off)

-l, --filter *filter-string*

Defines a filter to be applied to the scanned objects. See the section below on filters for more information.

-I, --subsystem-id *subsystem-id*

HPSS Subsystem Id (default=1).

-S, --process-shelf

Process files on shelved tapes.

-T, --process-trash

Process files in the trashcan.

-s, --quiet

Quiet mode - Do not display errors about skipped files.

-o, --output-dir *output-dir*

Where to write the files. Files will be written in the same directory structure as they are found in HPSS.

-R, --dry-run

Dry run mode. In this mode, everything is processed normally except that the generated batches are displayed instead of processing. There is no request for data movement in dry run mode.

-r, --recursive

Recurse directories. It should be noted that each subdirectory counts as an input source. The maximum number of input sources is \4096. If recursion results in more via directory listings, those directories will be skipped. (default=off)

-m, --data-directory *tmpdir*

Temporary directory for temporary input source data. Input source files are opened in an unlinked state, but still consume space on the file system until they are exhausted or lori ends.

-n, --thread-count *threads*

Work off the batches in multiple threads. The default thread count is one. Adding additional threads means that additional generated batches will be worked in parallel (see the dry run output for a list of the batches). Batches are generated such that the same volumes are always grouped together in contiguous batches to limit threading's potential for increasing the requested number of VVs. There is currently a limit of 2000 threads. Use caution when increasing the thread count and verify the HPSS environment is configured to support your workload. (default=1)

-N, --scan-thread-count *scan-threads*

Scan the inputs in parallel with the specified number of threads. (default=10)

-A, --enable-far-files *far-files*

Enable FAR stages if the number of files meets the provided threshold. (default=10)

-g, --disable-far

Disable FAR stages. (default=off)

-C, --complete-callback *callback-executable*

Specify a callback executable. In addition to reporting the read completion to stdout, lori will call the supplied executable after each read finishes. It is called with the following arguments:

arg1

Return Code

arg2

Request ID

arg3

BitfileID:BitfileHash

arg4

Caller UID

arg5

Caller GID

arg6

File Path

An example output is:

```
        0 d5983acccabb11e9aff45254000db6fb 0000347d-07-00000002-01e9a26392e7f4a4-
30e1:29186 0 0 /testfile
```

See tools/quaid/quaid_callout_example.ksh for an example callout script.



Since a process is forked for every successful read, the resulting process should be short lived. If longer processing is required, the callout should dump the results to an intermediary location to allow for consumption by other processes.

-x, --read-context *read-context-id*

Recover an existing read context using its identifier. This option ignores other input sources and instead uses a provided read queue context identifier to locate the read requests that need to be processed. This is useful for recovering read requests from a prior run of lori or some other tool which provides read queue context information.

-h, --help, -?, --usage

Display usage.

Additionally, lori responds to the HPSS_QUAID_MAX_FILES environment variable. This can be set either in the shell prior to invocation or in the HPSS env.conf file. This variable increases the maximum number of files that lori will allow to be processed before it begins rejecting additional files. The default is 10 million.

F.75.6. Examples

Example - read files in a directory

```
    #./lori -f /home/hpssuser
```

Example - read two directories and a file

```
    #./lori -f /home/hpssuser -f /home/hpssuser/project -f /home/hpssuser/file
```

Example - file list

```
    #./lori -F filelist
```

filelist format: /home/hpssuser /home/hpssuser/project

Example - filter by COS (COS == 1)

```
    #./lori -F filelist -lcos:1:eq
```

Example - only stage files which match a UDA (an attribute of /hpss/color set to *red*)

```
    #./lori -u '$DOC/hpss[color=red]'
```

Example - recover a read queue context

```
    #./lori -x 666353ce-4523-a549-88b5-e270902ca5fe
```

F.75.7. Output Format

Lori has two output streams: a results stream and an error stream.

The results stream is printed to stdout and contains information such as:

```
thread 0 running (batch 0)
SUCCESS:/a/test/file2
SUCCESS:/a/test/file3
ERROR (28):/a/test/file4
```

The "SUCCESS" entries are files which have been read. The message format:

```
SUCCESS:<filename>
```

```
ERROR (<error code>):<filename>
```

The error stream prints to stderr, and contains information not related to the read request, such as:

```
/a/test/file skipped - file is a duplicate
```

F.75.8. Filters

Filter format

Filters are specified in filter-name:value:operation format. For example, uid:300:eq is a filter which matches files whose uid attribute is equal to 300.

Available filter names

Filter Name	Description
family	File Family Id (ex: 1)
cos	Class of Service Id (ex: 1)
acct	Account Id (ex: 301)
uid	UID (ex: 301)
gid	GID (ex: 301)
Subsystem	Subsystem (ex: 1)
vv	PV Name (ex: V00001)
uda	User-defined Attribute (ex: $DOC/hpss[color='blue\'])



Only "eq" and "ne" operations are available for the UDA filter. XQuery provides finer-grained matching constructs.

Available filter operations

Operation Name	Operation Description
eq	Equal
ne	Not equal
gt	Greater than
ge	Greater than or equal
lt	Less than
le	Less than or equal

Limitations and Performance

Up to 20 filters may be specified. If ANY filter is matched, the file will be included in the input list (rather than having to match ALL filters).

Unlike the -u option, no index is needed for the UDA filter. However, it is still more resource-intensive than other filter types.

F.76. lscos(7)

F.76.1. Name

lscos - List and optionally delete files that have been earmarked for a change in class of service (COS).

F.76.2. Syntax

lscos [-S]
[-T]
[-d]
[-j]
[-i *sub_system_id*]
[-g *Global_DB_Name*]
[-k *Authenticator*]
[-m *Auth_Mechanism*]
[-p *Utility_Principal_Name*]
[-s *Database_Schema_Name*]

F.76.3. Description

"lscos" is an HPSS System Maintenance Utility. It is used to list the pathnames of files which have a change COS request pending. The utility outputs the list of records in the change COS table for the storage subsystem given on the command line. This can be useful if a user wants to find out if a COS change request has been submitted for a certain file. It also has an additional feature. A user can give the "-d" command line option, which will not only list the pending change COS records, but will delete them, cancelling the change COS request.

F.76.4. Parameters

-S

Provide a summary count of files earmarked for change COS by stream.

-T

Provide a detailed summary count of files earmarked for change COS by stream, family, and destination COS.

-d

Delete all pending change COS requests for the given subsystem. This option is not valid with -S or -T.

-i *Subsystem_ID*

Subsystem Id. The default value is taken from the environment variable HPSS_SUBSYS_ID.

-g *Global_DB_Name*

The global configuration database Name. The default value is taken from the environment variable HPSS_GLOBAL_DB_NAME.

-k *Authenticator*

The authenticator with which this utility will run. The default value is taken from the environment variable HPSS_PRIMARY_AUTHENTICATOR.

-m *Auth_Mechanism*

The security mechanism used to authenticate the client. The default value is take from the environment variable HPSS_PRIMARY_AUTHN_MECH.

-p *Principal_Name*

The server principal name under which this utility will run. The default value is taken from the environment variable HPSS_PRINCIPAL_SSM.

-s *Schema_Name*

The schema database Name. The default value is taken from the environment variable HPSS_MM_SCHEMA_NAME.

-j

Display the output in JSON format.

F.76.5. Example 1

The following command will list the pathnames of files with pending COS change requests for the local HPSS subsystem.

```
$ lscos -g cfg
FilesetRoot.3652:./scrub_test/test
```

The JSON output would be:

```
$ lscos -i 1 -g cfg -j
{
 "lscos": [
  {
   "FilesetName": "FilesetRoot.3652",
   "PathName": "./scrub_test/test3"
  }
 ]
}
```

F.76.6. Example 2

The following command will list the pathnames of files with pending COS change requests and cancel them.

```
$ lscos -d -i 1
FilesetRoot.3652:./scrub_test/test3...this change COS record is cancelled
```

F.76.7. Example 3

The following command will provide a count of pending COS change requests by stream.

```
$ lscos -i 1 -S
Stream ID            Count
0                    1
```

The JSON output would be:

```
$ lscos -i 1 -S -j
{
 "lscosSummary": [
  {
   "StreamID": 0,
   "Count": 1
  }
 ]
}
```

F.76.8. Example 4

The following command will provide a detailed summary count of files earmarked for change COS by stream, family, and destination COS.

```
$ lscos -i 1 -T
Stream ID  Family ID New COS ID            Count
        0          0          2                1
```

The JSON output would be:

```
$ lscos -i 1 -T
{
 "lscosSummaryDetail": [
  {
   "StreamID": 0,
   "FamilyID": 0,
   "NewCOSID": 3,
   "Count": 1
  }
 ]
}
```

F.76.9. Note

If the HPSS system consists of multiple storage subsystems and depending on the requested option, this utility may need to access metadata from one or more remote storage subsystem databases. Therefore, as a default, this utility must be invoked from the root storage subsystem machine (storage subsystem 1) for it to work properly. An alternative option is to manually catalog the needed storage subsystem database(s) on the desired machine where the utility will be invoked.

Invoking this utility with the -d option will result in changes to the HPSS metadata. Therefore, this utility must be invoked under a UNIX userid which has Db2 update privilege.

F.77. lsfilesets(7)

F.77.1. Name

lsfilesets - list all of the HPSS filesets.

F.77.2. Syntax

lsfilesets [-a *auth-type*] [-l] [-p *user*] [-k] [-t *keytab*] [-h] [-?]

F.77.3. Description

lsfilesets will list all the filesets in an HPSS system.

The user should be logged in as the HPSS Root User or as the SSM principal.

F.77.4. Parameters

-a *auth-type*
Used to specify the authentication type to use Supported values are: unix (Unix), krb5 (Kerberos)

-l
Used to create a new login credential (only valid for *unix* authentication)

-p
Used to specify a principal name

-k
Used to specify login using a keytab file

-t *keytab*
Used to specify the keytab file path

-h,-?
Display Usage

F.77.5. Example

The following command will list all of the filesets:

```
% lsfilesets
Fileset Name:     ball.archive
Fileset ID:       0.22
Gateway UUID:     2edf7f6a-def1-11d1-a574-02608c2cd22f
Core Server UUID: 5a475c92-deef-11d1-a574-02608c2cd22f
UID:              0
GID:              400
Class Of Service: 0
```

```
Family ID:        0
Directory Count:  55
File Count:       18
Hard Link Count:  0
Sym Link Count:   0
Junction Count:   0
Fileset Type:     ARCHIVED

Fileset Name:     DebbieTest2
Fileset ID:       3272205073.4240182136
Gateway UUID:     00000000-0000-0000-0000-000000000000
Core Server UUID: 5a475c92-deef-11d1-a574-02608c2cd22f
UID:              4897
GID:              400
Class Of Service: 0
Family ID:        0
Directory Count:  1
File Count:       0
Hard Link Count:  0
Sym Link Count:   0
Junction Count:   0
Fileset Type:     HPSS_ONLY

Number of Filesets Found:   2
```

The following command will list all of the filesets using unix authentication, a default keytab, and the ssm principal.

```
% lsfilesets -a unix -k -p hpssssm

Fileset Name:     FilesetRoot.2137
Fileset ID:       3272711636.4289611120  (0xc3119dd4ffae4570)
Gateway UUID:     00000000-0000-0000-0000-000000000000
Core Server UUID: 92b4cec6-2866-11de-8a2a-00195b713adf
UID:              0
GID:              0
Class Of Service: 0
Family ID:        0
Directory Count:  253
File Count:       4189271
Hard Link Count:  4
Sym Link Count:   3
Junction Count:   1
Fileset Type:     HPSS_ONLY

Fileset Name:     Fileset 2121849828
Fileset ID:       3995083612.1403019277  (0xee20235c53a0600d)
Gateway UUID:     00000000-0000-0000-0000-000000000000
Core Server UUID: 92b4cec6-2866-11de-8a2a-00195b713adf
UID:              0
```

```
        GID:              0
        Class Of Service: 0
        Family ID:        0
        Directory Count:  1
        File Count:       1
        Hard Link Count:  0
        Sym Link Count:   0
        Junction Count:   0
        Fileset Type:     HPSS_ONLY

        Number of Filesets Found:   2
```

F.78. lshpss(7)

F.78.1. Name

lshpss - List information about HPSS

F.78.2. Syntax

lshpss [options ...]

F.78.3. Description

lshpss reads HPSS configuration metadata directly from the database, and displays summary information in a formatted form on standard output.

F.78.4. Options

-global
Show Global Configuration Data

-subsys
Show Storage Subsystems

-cos
Show Class of Service list

-cosdet
Show Class of Service Detail list

-hier
Show Hierarchy list

-sc
Show Storage Class list

-migp
Show Migration Policies

-purgep
Show Purge Policies

-dev
Show Mover Devices

-drv
Show PVL Drives

-svr

Show HPSS Servers

-mvr

Show HPSS Movers

-rait

Show RAIT Servers

-acct

Show Accounting Policies

-logp

Show Log Policies

-fset

Show Filesets

-ffam

Show File Families

-core

Show Core Servers

-pvr

Show PVRs

-gate

Show Gatekeeper

-mpsvr

Show Migration/Purge Server

-all

Display all of the above

-vol

Show PVL Physical Volumes

-g *global database name*

The name of the global database. If not present, the value for this parameter will be taken from the environment variable HPSS_GLOBAL_DB_NAME.

-s *database schema name*

The schema name associated with the HPSS Databases. If not present, the value for this parameter will be taken from the environment variable HPSS_MM_SCHEMA_NAME.

-h

Show this help message

-l

Squelch lshpss header

-j

Output in JSON format

-since *timestring*

Output the values at the given time (must be after enabling history) The time string format is YYYY-MM-DD-HH.MM.SS.nnnnnnnnnnnn. The nanoseconds are optional.

F.78.5. Example

The following command will list all the resources :

```
$ lshpss -all
```

To list classes of service and purge policies :

```
$ lshpss -cos -purgep
```

To list classes of service and purge policies as they were on January 29, 2025:

```
$ lshpss -cos -purgep -since 2025-01-29-14.39.33.01
```

F.79. lsjunctions(7)

F.79.1. Name

lsjunctions - list all of the junctions that are being managed by the Core Server.

F.79.2. Syntax

lsjunctions [-a *auth-type*] [-l] [-p *user*] [-k]
[-t *keytab*] [-i *subsystem id*]

F.79.3. Description

lsjunctions will list all the junctions, and their associated filesets, that are being managed by the Core Server. Since multiple junctions can be linked to a single fileset, it is possible to have similar listings (except for the junction name) for different junctions that point at the same fileset.

The utility may only be run by a root user.

F.79.4. Parameters

-a

Used to specify the authentication type to use

1. Supported values are: unix (Unix), krb5 (Kerberos)

-l

Used to create a new login credential

-p

Used to specify a principal name

-k

Used to specify login using a keytab file

-t *keytab*

Used to specify the keytab file path

-i *subsystem id*

Used to specify the subsystem id

F.79.5. Example

The following command will list out all the junctions.

```
        % lsjunctions

        Junction Name:    FilesetRoot.2546: ./home/hpss/fs_hpss
        Fileset Name:     File Family FS
```

```
Fileset ID:       2178027353.1221831000
UID:              0
GID:              400
Gateway UUID:     00000000-0000-0000-0000-000000000000
Core Server UUID: eff42414-8de6-11d1-9712-08005abaaca9
Class Of Service: 8
Family ID:        300
Fileset Type:     HPSS_ONLY

Junction Name:    FilesetRoot.2546: ./home/hpss/fs_6a139
Fileset Name:     fs_6a139
Fileset ID:       0.139
UID:              2839
GID:              400
Gateway UUID:     434d8292-8de5-11d1-9712-08005abaaca9
Core Server UUID: eff42414-8de6-11d1-9712-08005abaaca9
Class Of Service: 5
Family ID:        0
Fileset Type:     ARCHIVED

Junction Name:    FilesetRoot.2546: ./home/hpss/empty_junction
Fileset Name:     <No corresponding fileset>

Number of Junctions Found:   3
```

The following command will list out all the junctions on subsystem 1 using unix authentication, the default keytab, and a trusted user.

```
% lsjunctions -a unix -k -p hpssssm -i 1
Junction Name:    FilesetRoot.2137: ./tmp/junc
Fileset Name:     Fileset 2121849828
Fileset ID:       3995083612.1403019277
UID:              0
GID:              0
Gateway UUID:     00000000-0000-0000-0000-000000000000
Core Server UUID: 92b4cec6-2866-11de-8a2a-00195b713adf
Class Of Service: 0
Family ID:        0
Fileset Type:     HPSS_ONLY

Number of Junctions Found: 1
```

F.79.6. See Also

crtjunction, deljunction, and lsfilesets

F.80. lspvhist(7)

F.80.1. Name

lspvhist - List tape physical volume history information

F.80.2. Syntax

lspvhist [--subsystem-id=*ss*]
[--authentication-mechanism=*am*]
[--authenticator-name=*an*]
[--db-schema-name=*sn*]
[--global-db-name=*dn*]
[--login-principal=*lp*]
[--pv-name=*pv*]
[--select-device-id=*di*]
[--select-device-name=*dn*]
[--select-drive-address=*da*]
[--select-error-code=*ec*]
[--select-mover-name=*mn*]
[--select-operation=*op*]
[--vv-name=*vv*]
[--json]

or

lspvhist [-i *ss*]
[-m *am*]
[-k *an*]
[-s *sn*]
[-g *dn*]
[-p *lp*]
[-n *pv*]
[-D *di*]
[-N *dn*]
[-d *da*]
[-e *ec*]
[-M *mn*]
[-o *op*]
[-v *vv*]
[-j]

F.80.3. Description

"lspvhist" is a HPSS System Maintenance Utility. It is used to list the information that is kept about changes that have been made to tape physical volumes (PVs), I/O errors that have occurred and reclaim events. When any of these events occur, the Core Server records the circumstances of the event as part of the PV metadata.

The information includes the Virtual Volume ID (VVID), the PV Name, the date and time, the operation that was attempted or completed, the error code, the name of the tape mover that was involved, the device ID, name and drive address on which the tape was mounted, the drive media type, the relative address at which the error occurred, the expected and actual transfer lengths and a comment.

Eight different events are logged:

1. Tape read errors
2. Tape write errors
3. Tape mount errors
4. VV Condition changes
5. PV Condition changes
6. Tape position errors
7. Tape write tape mark errors
8. VV Reclaims

For each of these events, different combinations of the information fields described above are recorded. Some events, such as VV and PV changes, record only the PV or VV ID, date, time and a comment. Other events, such as Tape Write Errors, record all of the fields. The program displays all of the information available for each event, but does not display fields that contain no information.

This program became part of HPSS in version 7.4.1. This first version provides a basic service. It provides a number of selection criteria which are logicalled "anded" together, selects the rows from the database that match these criteria, formats them for display and sends the result to standard output. If no rows match the selection criteria, nothing is output.

Certain combinations of selection criteria will return history entries unsorted — not in ascending chronological order.

F.80.4. Options

--authentication-mechanism=*am*,-m *am*

The security mechanism used to authenticate the client. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--authenticator-name=*an*,-k *an*

The authenticator which is used to obtain credentials. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--db-schema-name=*sn*,-s_sn_

The ASCII name of the Db2 schema that contains the global database. If not specified on the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--global-db-name=*dn*,-g *dn*

The ASCII name of the global database where the basic configuration information for HPSS is

found. If not specified on the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--login-principal=*lp*,-p *lp*

The server principal name which this utility will run under. Lsvol is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided on the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--pv-name=*pv*,-n *pv*

Limits the output to history entries associated with the named Physical Volume. If the PV is part of a striped VV, history information about the other PVs in the VV is not returned.

--select-device-id=*di*, -D *di*

Limits the output to history entries associated with the given tape device ID.

--select-device-name=*dn*, -N *dn*

Limits the output to history entries associated with the named tape device. Note that the name given in the command line option must exactly match the names in the history entries for them to be returned.

--select-drive-address=*da*, -d *da*

Limits the output to history entries associated with the named tape drive address. Note that the address given in the command line option must exactly match the addresses in the history entries for them to be returned.

--select-error-code=*ec*, -e *ec*

Limits the output to history entries that contain the given error code. Note that most HPSS error codes are negative numbers.

--select-mover-name=*mn*, -M *mn*

Limits the output to history entries associated with the named mover. Note that the name given in the command line option must exactly match the names in the history entries for them to be returned.

--select-operation=*op*, -o *op*

Limits the output to history entries that contain the given operation code. The operation codes are:

```
Read Error             1
Write Error            2
Mount Error            3
VV Change              4
PV Change              5
Position Error         6
Write Tape Mark Error  7
VV Reclaim             8
```

--subsystem-id=*ss*, -i *ss*

The number of the sub-system that contains the Core Server of interest. This argument must be a positive non-zero value. If not provided on the command line, the value of the environment variable HPSS_SUBSYS_ID will be used.

--vv-name=*vv*, -v *vv*

Limits output to history entries associated with the named Virtual Volume. You may use the name of any of the PVs in the VV to select the VV. When history entries are selected by VV, entries associated with all of the PVs in the VV will be returned.

Note that if a VV is reclaimed, all history entries associated with prior instances of the VV are not accessable with this option.

--json, -j

Output in JSON format.

F.80.5. Examples

Select the history entries for all PVs in the system:

```
% lspvhist

<The entire database is returned, which may be quite large>
```

Select the history entries for all PVs in a given VV:

```
% lspvhist --vv-name=UC7028

PV Name:                      UC7028
Time:                         Wed Feb  8 15:42:02 2012
Operation:                    Tape read error
Error:                        -5
Mover Name:                   Tape Mover (azkaban1)
Device ID:                    2000
Device Name:                  /dev/lst0
Device Type:                  3580 Gen3 Tape
Drive Address:                1,0,1,0
Relative Address:             Section: 3, Offset: 0
Expected transfer length:     4194304
Actual transfer length:       2097152

PV Name:                      UC7028
Time:                         Wed Feb  8 15:59:21 2012
Operation:                    Tape read error
Error:                        -5
Mover Name:                   Tape Mover (azkaban1)
Device ID:                    2000
Device Name:                  /dev/lst0
```

```
Device Type:                3580 Gen3 Tape
Drive Address:              1,0,1,0
Relative Address:           Section: 3, Offset: 0
Expected transfer length:   4194304
Actual transfer length:     2097152

...
```

Select the history entries associated with a given tape device:

```
% lspvhist --select-device-id=2012

PV Name:                    UC7089
Time:                       Thu Feb 16 16:19:12 2012
Operation:                  Tape write error
Error:                      -5
Mover Name:                 Tape Mover (azkaban1)
Device ID:                  2012
Device Name:                /dev/lst12
Device Type:                3580 Gen3 Tape
Drive Address:              1,3,1,0
Relative Address:           Section: 7, Offset: 0
Expected transfer length:   33554432
Actual transfer length:     33554432

PV Name:                    UC7086
Time:                       Wed Feb  8 17:36:56 2012
Operation:                  Tape write error
Error:                      -5
Mover Name:                 Tape Mover (azkaban1)
Device ID:                  2012
Device Name:                /dev/lst12
Device Type:                3580 Gen3 Tape
Drive Address:              1,3,1,0
Relative Address:           Section: 1, Offset: 0
Expected transfer length:   33554432
Actual transfer length:     33554432

PV Name:                    UC7092
Time:                       Wed Feb  8 17:38:32 2012
Operation:                  Tape write error
Error:                      -5
Mover Name:                 Tape Mover (azkaban1)
Device ID:                  2012
Device Name:                /dev/lst12
Device Type:                3580 Gen3 Tape
Drive Address:              1,3,1,0
Relative Address:           Section: 1, Offset: 0
Expected transfer length:   33554432
```

```
      Actual transfer length:      33554432
```

Select the history entries for all PVs that changed Condition:

```
      % lspvhist --sub=1 --select-op=5

      PV Name:                     UC7089
      Time:                        Thu Feb 16 16:19:12 2012
      Operation:                   Change PV
      Comment:                     Set to RO by the server

      PV Name:                     UC7092
      Time:                        Fri Mar  2 15:29:35 2012
      Operation:                   Change PV
      Comment:                     Set to DOWN by the Admin
```

F.81. lsrb(7)

F.81.1. Name

lsrb - List of files, grouped by tape volume and sorted by date, that have a last access date prior to a specified date.

F.81.2. Syntax

lsrb *yyyymmdd* [-p *Utility Principal Name*]
[-k *Utility Authenticator*]
[-g *HPSS Global Config Database*]
[-m *Security Mechanism*]
[-s *HPSS Database Schema Name*]
[-S *Sleep Value (seconds)*]

F.81.3. Description

LSRB utility is a HPSS system maintenance utility. It is used to generate a list of files on tape that have a last access date prior to the specified date.

The list of files will be grouped by tape volume, and sorted in ascending order (oldest file first). This can be useful when identifying files that are candidates for deletion because they have not been accessed for a long time. A date must be provided to the LSRB utility in the yyyymmdd format (e.g. 19980131 — January 31, 1998)).



This tool only scans tape volumes. It does not scan disk volumes.

F.81.4. Options

-p *Principal Name*

The server principal name which this utility will run under. Usually run as SSM. If not present, the value for this parameter will be taken from the HPSS_PRINCIPAL_SSM environment variable.

-k *Authenticator*

The authenticator which this utility will run with. Usually run as SSM. If not present, the value for this parameter will be taken from the HPSS_PRIMARY_AUTHENTICATOR environment variable.

-g *Global Cfg DB*

The name of the Db2 database containing the global configuration information. If not present, the value for this parameter will be taken from the HPSS_GLOBAL_DB_NAME environment variable.

-m *Security Mechanism*

Used to specify the authentication type to use. Supported values include: unix (Unix authentication), krb5 (Kerberos)

-s *Schema Name*

The name of the Db2 schema for HPSS metadata. If not present, the value for this parameter will be taken from the HPSS_MM_SCHEMA_NAME environment variable.

-S *Sleep Value*

The Sleep Value is used to throttle back the utility. It represents the number of seconds to sleep between physical volumes to reduce the utility's load on the CPU. If not present, the value for this parameter will be taken from the HPSS_LSRB_SLEEP environment variable (default=0).

F.81.5. Example

The following command will list files on tape that were accessed prior to January 01, 1996. The list of files will be sorted and grouped by volume.

```
lsrb 19960101 -p hpssssm -k /var/hpss/etc/hpss.unix.keytab \
              -g cfg -s hpss
```

F.81.6. See Also

lsvol

F.82. lsvol(7)

F.82.1. Name

lsvol - List pathnames of files with segments on a volume.

F.82.2. Syntax

lsvol --vv-name=*vv*
[--subsystem-id=*ss*]
[--audit-volume]
[--authentication-mechanism=*am*]
[--authenticator-name=*an*]
[--db-schema-name=*sn*]
[--global-db-name=*dn*]
[--login-principal=*lp*]
[--lstapes-format]
[--extractor-format]
[--bitfile]
[--json]
[--report-orphans]
[--bucket-name-like=*bn*]

lsvol -v *vv*
[-i *ss*]
[-a]
[-m *am*]
[-k *an*]
[-s *sn*]
[-g *dn*]
[-p *lp*]
[-t]
[-e]
[-b]
[-j]
[-o]
[-n *bn*]

F.82.3. Description

"lsvol" is a HPSS System Maintenance Utility. It is used to list the pathnames of files which have one or more storage segments on the given Physical Volume. This can be useful when a cartridge is damaged and a list of files affected is needed.

If certain command line arguments are not specified, HPSS environment variables will be used.

F.82.4. Options

--audit-volume, -a

Audit tapes. When this option is specified the program produces a line of output for each tape storage segment on the tape. If a file is made up of more than one storage segment, it will appear in the output once for each segment. The default behavior is to list each file only once.

--authentication-mechanism=*am*, -m *am*

The security mechanism used to authenticate the client. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--authenticator-name=*an*, -k *an*

The authenticator which is used to obtain credentials. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--db-schema-name=*sn*, -s *sn*

The ASCII name of the Db2 schema that contains the global database. If not specified on the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--global-db-name=*dn*, -g *dn*

The ASCII name of the global database where the basic configuration information for HPSS is found. If not specified on the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--login-principal=*lp*, -p *lp*

The server principal name which this utility will run under. Lsvol is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided on the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--lstapes-format, -t

Generate lstapes format output. This option is limited to the listing of tapes only.

The format is as follows, in space-delimited format: - Bitfile Id - Storage Class Id - Date/Time of Last Write - PV Name:Section Number+Section Offset,Written Length,Bitfile Offset+Bitfile Segment Length + Storage Segment Offset - File Object Unix Mode - File Object UID - File Object GID - File Object Data Length - File Object Time Last Written - File Object Path (from last junction)

--extractor-format, -e

Generate extractor format output. This option is limited to the listing of tapes only.

The format is as follows, in space-delimited format: - Bitfile Id - Storage Class Id - Date/Time of Last Write - PV Name:Section Number+Section Offset,Written Length,Bitfile Offset+Bitfile Segment Length + Storage Segment Offset - File Object Unix Mode - File Object UID - File Object GID - File Access Time - File Modified Time - File Link Count - File Object Data Length - File Object Path (from last junction)

--bitfile, -b

Do not do path name lookups, instead print bitfile IDs and distribution keys. This option is not compatible with the extractor or lstapes format.

--json, -j

Display the output in json format.

--report-orphans, -o

Report the storage segment ID for files that did not list for a tape volume. The segment may be in one of three states.

- In Bitfile Checkpoint table
- In Tape Segment Unlink table
- Orphaned segment

Segments in the Bitfile Checkpoint or Tape Segment Unlink tables are pending deletion. If lsvol is running for a while over a large amount of files, the segment IDs reflect the state at the end of lsvol execution.

--subsystem-id=*ss*, -i *ss*

The ID of the subsystem that contains the Core Server of interest. This argument must be a positive non-zero value. If not specified on the command line, the value of the HPSS_SUBSYS_ID environment variable will be used.

--vv-name=*vv*, -v *vv*

The name of the Virtual Volume to list. The name may be 6 or 8 characters long. If 6 characters are provided, lsvol will add two zeros to the name to obtain the conventional HPSS 8 character name. You may use the name of any PV in the VV to identify the VV.

--bucket-name-like=*bn*, -n *bn*

Display the contents of a bucket, or buckets names matching the specified bucket name. Wild-card characters are specified with an asterisk (*).

F.82.5. Example

The following command will list the pathnames of files with segments on the physical volume 865AFN00.

```
$ lsvol --vv-name=865AFN
    28 ./home/hpss/subsys1_large_pftp/water_91392_cos45_21
     4 ./home/hpss/subsys1_large_pftp/water_45018_cos45_8
    24 ./home/hpss/subsys1_large_pftp/water_91392_cos45_6
    19 ./home/hpss/subsys1_large_pftp/water_104962_cos45_12
    33 ./home/hpss/subsys1_large_pftp/water_51916_cos45_18
    30 ./home/hpss/subsys1_large_pftp/water_51916_cos45_4
    32 ./home/hpss/subsys1_large_pftp/water_51916_cos45_7
    23 ./home/hpss/subsys1_large_pftp/water_91392_cos45_5
    35 ./home/hpss/subsys1_large_pftp/water_51916_cos45_21
```

```
 9 ./home/hpss/subsys1_large_pftp/water_45018_cos45_21
11 ./home/hpss/subsys1_large_pftp/water_45018_cos45_24
20 ./home/hpss/subsys1_large_pftp/water_104962_cos45_25
26 ./home/hpss/subsys1_large_pftp/water_91392_cos45_18
15 ./home/hpss/subsys1_large_pftp/water_104962_cos45_6
 3 ./home/hpss/subsys1_large_pftp/water_45018_cos45_5
25 ./home/hpss/subsys1_large_pftp/water_91392_cos45_8
17 ./home/hpss/subsys1_large_pftp/water_104962_cos45_8
 2 ./home/hpss/subsys1_large_pftp/water_45018_cos45_1
10 ./home/hpss/subsys1_large_pftp/water_45018_cos45_22
12 ./home/hpss/subsys1_large_pftp/water_45018_cos45_25
14 ./home/hpss/subsys1_large_pftp/water_104962_cos45_4
 1 ./home/hpss/subsys1_large_pftp/water_101732_cos45_1
22 ./home/hpss/subsys1_large_pftp/water_91392_cos45_4
 5 ./home/hpss/subsys1_large_pftp/water_45018_cos45_11
 7 ./home/hpss/subsys1_large_pftp/water_45018_cos45_16

Number of File Pathnames found: 25
Number of errors getting File Pathnames: 0
Number in Bitfile Checkpoint table: 0
Number in Tape Segment Unlink table: 0
Number of orphaned segments: 0
```

F.82.6. See Also

dumpbf, dumppv_pvr, dumppv_pvl

F.83. lsvol_graph(7)

F.83.1. Name

lsvol_graph - Output a visualization of a tape volume

F.83.2. Syntax

lsvol_graph --file *FILE*
[--output *OUTPUT*]
[--title *TITLE*]
[--group *GROUP*]
[--show-paths]
[--colors]
[--dont-merge]
[--help]

F.83.3. Prerequisites

This tool has a few direct pre-requisite python packages. All of these can be installed via pip. Each package has multiple dependent packages, so the initial pre-req gathering may take a while.

The direct pre-requisites are:

- Plotly (for graphing)
- Plotly Resampler (improves graphing performance)
- Pandas (data visualization tool)
- kaleido (for outputting images)

F.83.4. Description

This tool takes lsvol json output for a tape volume and turns it into a plotly graph. This is useful to visualize a tape volume to identify patterns or issues that may be difficult to notice from the lsvol text output. For example, identifying large empty sections, small file migration issues, poorly co-located data, etc.

The graph displays the various files listed by lsvol in a stacked bar chart. By default, this is done by file "type" (aggregate, single file). However it can also be grouped by other criteria such as UID or Parent directory.

This tool should not be taken as having authoritative information about tape positioning information for the purpose of reading from a tape.

F.83.5. A caveat on tape marks

HPSS does not track empty space or tape marks directly. The tool attempts to identify tape mark data based on gaps on the tape. If the gap looks like a tape mark, then it is marked as such.

Otherwise, the gap is considered empty space. Empty space can occur because data was deleted from that area after it was written, or was perhaps rolled back after some prior write operation. Also note that because we are inferring the gaps based on the lsvol output, we do not show internal tape marks since there is no lsvol metadata that would allow us to identify them.

F.83.6. Options

-o OUTPUT, --output OUTPUT

This controls the output location. By default, lsvol_graph will open a browser window to an interactive graph. By specifying the -o option, you can instead output the graph to a scalable vector graphics file (.svg). Note that this output is not interactive.

For example, -o /tmp on myvol.json will write MYVOL0.svg to /tmp.

-p, --show-paths

Show paths on the tape (default: False)

-d, --dont-merge

By default, the graphing tool merges **contiguous** sections which have the same grouping type (for example, two single files , two aggregate files) because for very large tapes this can result in a 1000x less processing. In some cases, it may be desirable not to merge these sections.

Don't merge common types (can dramatically slow rendering) (default: False)

-c COLORS, --colors COLORS

Define custom colors. This works differently depending on the grouping mechanism selected.

The "type" colors must be individually and exhaustively identified. For example, this will make single files yellow, aggregates brown, empty space black, and tape marks white.

```
-c '{"File": "yellow", "Aggregate": "brown", "Empty": "black", "Tape Marks": "white"}'
```

Note the single quotes around the argument.

For the other groupings (e.g. UID, Parent), the color is defined using plotly color spectrum names. The default type is "Inferno". Plotly has a number of colorscales that are built in; see the Plotly documentation for a fuller discussion. For example:

```
-c Inferno
```

would give the default color scheme

-g GROUP, --group GROUP

How to group data (Type, Parent, UID) (default: Type)

The "Type" is the way that the data has been written to tape. It's either a single file, aggregate file, empty, or a tape mark.

The "Parent" is the parent path, based on the file path. Empty sections and Tape Marks are still identified as such.

The "UID" is the file owner’s UID.

F.83.7. Example

First, select a volume you want to analyze. Having the volume, use lsvol with the -t and -j options to generate a detailed output of the tape volume contents and redirect that to a file.

% lsvol -v MYVOL0 -t -j > myvol.json

Next, run it through the lsvol_graph tool:

% lsvol_graph -f myvol.json

This will create a graph based on the file types (aggregate, single file) for the myvol.json file.

F.83.8. See Also

lsvol

F.84. manage_temporal(7)

F.84.1. Name

manage_temporal.py - Manages the HPSS history tables

F.84.2. Synopsis

manage_temporal.py --help
[--action=*ACTION*]
[--instance_name=*DBINSTANCE*]
[--database_name=*DBNAME*]
[--user_name=*DBUSER*]
[--password=*DBPASS*]
[--schema_name=*TABSCHEMA*]
[--table_name=*TABLE*]
[--partition_name=*DPNAME*]
[--low_month=*MONTH*]
[--low_year=*YEAR*]
[--nosavedata]
[--csv_file=*FNAME*]

F.84.3. Description

manage_temporal.py is a Python utility for adding, culling, and enumerating history tables in the HPSS databases. History tables are necessary for Rumbler usage, where change tracking of the HPSS filesystem can be reported as a series of JSON records for easier parsing by 3rd parties such as global namespaces.

F.84.4. Options

--help prints a summary of options.

ACTION must be one of:

list_removable

Lists the tables which can be detached. If combined with the **--csv_file** option, lists all history tables, partitions, and indexes, as well as detachable and non-detachable tables. Each type will be written to its own CSV file with a related suffix appended.

list_partitions

Lists and prints basic information about all data partitions.

list_info

Lists and prints basic information about all existing indices.

filter_partitions

Lists all partitions which match the given filter criteria.

detach_partition

Removes a set of partitions from the active database history.

add_partition

Adds a set of partitions to the active database history.

list_idxpartitions

Lists the partition index tables

--instance_name=*DBINSTANCE*

Name of the database instance, default is read from configuration

--database_name=*DBNAME*

Name of the database to connect to, default is read from configuration

--user_name=*DBUSER*

User name to connect to database with, default is read from mm.keytab (or unset if it does not exist)

--password=*DBPASS*

Password to connect to database with, default is read from mm.keytab (or unset if it does not exist). **Note** it is a bad idea to pass a password via the command line.

--schema_name=*TABSCHEMA*

schema name of the partitioned hpss history table, default: HPSS

--table_name=*TABLE*

partitioned hpss history table name
TABLE must be one of NSOBJECT_HISTORY, BITFILE_HISTORY, NSACL_HISTORY, NSFILESETATTR_HISTORY, NSTEXT_HISTORY, NSTRASH_HISTORY, or USERATTRS_HISTORY
default: NSOBJECT_HISTORY

--partition_name=*DPNAME*

Partition name to detach from the table, default: None
If a partition name is given, that exact partition will be manipulated. Otherwise a filtered range will be manipulated based on these following two options.

--low_month=*MONTH*

Numerical month of low date range of partition, defaults to current integer month (i.e. January = 1). Range 1-12

--low_year=*YEAR*

4-digit year of low date range of partition, defaults to current year. Range 2022-2052

--nosavedata

Do not save history partition in a table, default: Save detached data into a table

--csv_file=*FNAME*

Output partition data to a csv file, default: None
There will be a set of files created, each beginning with the given FNAME and with its category type appended.

F.84.5. Initial Installation

As a prerequisite, the HPSS Python CLAPI must be installed.

Two main tasks must be completed before using **manage_temporal**. First, the Python package ibm_db must be installed and set up to use the instance database. Second, the UDF routines must be added to the databases.

F.84.5.1. Installing the Python Database Interface Package

The easiest way to install is with Python's PIP installer. On Redhat, the PIP installer may need to appended with the Python version, ie **pip3**. Run the following commands either as root or as the user that will be performing the database maintenence:

```
export IBM_DB_HOME=/opt/ibm/db2/default
pip3 install ibm_db
```

F.84.5.2. Adding libdb2.so.1 to the Library Search Path

A file must be created in /etc/ld.so.conf.d which lists the DB2 installation directory's lib64 subdirectory. If you created a link from /opt/ibm/db2/default to /opt/ibm/db2/<installed-version>, you could use this link as the base path.

```
echo "/opt/ibm/db2/default/lib64" > /etc/ld.so.conf.d/ibm-db2.conf
ldconfig
```

F.84.5.3. Installing the UDF Routines

If the system has been upgraded at any time by QREP, the routines should already be installed. The UDF routines must otherwise be installed by the administrator. There is a script in the *tools* directory to accomplish this.

```
cd /opt/hpss/tools/metadata/udf
./udfcreate.sh -d hsubsys1 -l /opt/hpss/lib/libdb2_hpssutils.so
./udfcreate.sh -d hcfg -l /opt/hpss/lib/libdb2_hpssutils.so
```

F.84.6. Caveats

The manage_temporal utility will only work on partitions for one table at a time. Therefore, full maintenance requires that the utility be run once for every history table, for every planned operation. This shortcoming is to be addressed by future work.

F.84.7. Author

Daniel Sands, Sandia National Labs, Albuquerque, NM

F.85. hpss_add_temporal_tables(7)

F.85.1. Name

hpss_add_temporal_tables.py - Add HPSS Temporal Tables

F.85.2. Synopsis

hpss_add_temporal_tables.py [--database=*DBNAME*]
[--ddldir=*DDLDIR*]
[--instance=*INSTANCE*]
[--schema=*SCHEMA*]
[--months=*MONTHS*]
[--dbuser=*DBUSER*]
[--dbgroup=*DBGROUP*]
[--verbose]
[--dry-run]

F.85.3. Description

hpss_add_temporal_tables.py is a Python utility for adding temporal tables to an HPSS database.

F.85.4. Options

--database:=*DBNAME*

Name of the database to connect to. Must be supplied.

--ddldir:=*DDLDIR*

DDL directory, default is /opt/hpss/config/db2scripts.

--instance:=*INSTANCE*

The Db2 instance owner, default is hpssdb.

--schema:=*SCHEMA*

The database schema, default is hpss.

--months:=*MONTHS*

The number of months for range partitioning, default is 24 months and must be ⇐ 72 months and >= 0 months.

--dbuser:=*DBUSER*

The user to grant query privileges to, default is hpss.

--dbgroup:=*DBGROUP*

The group to grant query privileges to, default is hpsssrvr.

--verbose

Verbose print the output of every command.

--dry-run

Run without executing any commands.

F.86. hpss_remove_temporal_tables(7)

F.86.1. Name

hpss_remove_temporal_tables.py - Remove HPSS Temporal Tables

F.86.2. Synopsis

hpss_remove_temporal_tables.py [--database=*DBNAME*]
[--schema=*SCHEMA*]
[--instance=*INSTANCE*]
[--verbose]

F.86.3. Description

hpss_remove_temporal_tables.py is a Python utility for removing temporal tables from the HPSS database.

F.86.4. Options

--database:=*DBNAME*

Name of the database to connect to. Must be supplied.

--schema:=*SCHEMA*

The database schema, default is hpss.

--instance:=*INSTANCE*

The Db2 instance owner, default is hpssdb.

--verbose

Print out everything that will be done.

F.87. mkmprec(7)

F.87.1. Name

mkmprec - Make Migration or Purge Records

F.87.2. Syntax

mkmprec -t *Storage Class ID*
-r (*m* - migration, *p* - purge or *b* - both)
[-i *Storage Subsystem ID*]
[-n *Core Server Description Name*]
[-g *HPSS Global DB Name*]
[-p *Utility Principal Name*]
[-k *Utility Authenticator Name*]
[-m *Utility Security Mechanism*]
[-s *Schema Name*]

F.87.3. Description

"mkmprec" is an HPSS System Maintenance Utility. It may be used to generate migration or purge records. It generates records by writing directly to metadata. Because of this, it should be run during a scheduled maintenance period while core HPSS servers have been shutdown. The utility will scan through the BFS disk map entries for those of the specified storage class and subsystem and generate purge/migrate records as required. Migration, purge or both types of records can be generated and are specified via input argument.

A situation when this utility is needed is the addition of a another level to the hierarchy to an existing class of service. Migration records will not be created for the storage class at the bottom of the hierarchy. If another storage class is added to the bottom of an existing hierarchy, the files existing previously to the addition will not have associated migration records. This utility can be invoked during a maintenance period to create migration records.

F.87.4. Options

-i *Storage SubsysID*

The Storage Subsystem ID on which to operate. The default value of this argument is the value of system environment variable HPSS_SUBSYS_ID.

-p *Principal Name*

The server principal name which this utility will run under, usually SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. The default value of this argument is the value of system environment variable HPSS_PRINCIPAL_SSM.

-k *Authenticator*

The authenticator with which this utility will run. The default value of this argument is the value of the system environment variable HPSS_PRIMARY_AUTHENTICATOR.

-m *Mechanism*

The security mechanism which this utility will use to authenticate the client. The default value of this argument is the value of the system environment variable HPSS_PRIMARY_AUTHN_MECH.

-g *Global Cfg*

The global database name with which this utility will run. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

-s *Schema Name*

The HPSS database schema name with which this utility will run. If set, the environment variable HPSS_MM_SCHEMA_NAME will be used if an argument is not specified on the command line.

-t *Storage Class ID*

The storage class id of the volumes to be retired. The Storage Class id can be found from SSM through: Monitor, Storage Class Lists, Active screen.

-r *Record Type*

The type of record to create. Legal values are: m (migration records are created), p (purge records are created), b (both migration and purge records are created).

-n *Core Server Name*

The core server descriptive name. If this is not specified, the value of the environment variable HPSS_DESC_CORE is used, or its default value "Core Server."

F.87.5. Example

The following command will generate migration records for files in storage class 100, subsystem 1:

```
$ mkmprec -i 1 -g cfg -s HPSS -k /var/hpss/etc/hpss.keytab
          -m krb5 -p hpss_ssm -t 100 -r m
```

Or using the default enviroment variables:

```
$ mkmprec -t 100 -r m
```

F.88. mps_reporter(7)

F.88.1. Name

mps_reporter - HPSS Migration/Purge Server Report Utility

F.88.2. Syntax

mps_reporter [-h] [-b | -e | -E] [-c *STORAGE_CLASS_ID*] [-H] [-s]
[--debug] [-v]

mps_reporter [-h] [-b | -e | -E] [-c *STORAGE_CLASS_ID*] [-H]
[-f *FILE*] [--debug] [-v]
[*FILE* [*FILE* ...]]

F.88.3. Description

mps_reporter is a utility which generates user-readable output from raw report files created by the Migration/Purge Server (MPS).

Each file in <FILE> is processed and the output is sent to stdout.

A report contains the following types of information based on events recorded in the MPS:

When the MPS started up.

Information about the disk migration, disk purge, or tape migration of a bitfile.

Summary information of a disk migration, disk purge, or tape migration.

F.88.4. Options

-h, --help
Show this help message and exit

-b, --brief
Generate a brief summary report only (default: False)

-e, --output-only-error
Only output error entries (default: False)

-E, --suppress-error
Suppress error entries (default: False)

-c *STORAGE_CLASS_ID*, --storage-class-id *STORAGE_CLASS_ID*
Only output migration and purge entries related to the specified storage class (default: None)

-H, --with-file-name
Print the file name for each line of output; this is the default when there is more than one file

specified (default: False)

-s, --stdin

Read from stdin (default: False)

-f *FILE*, --file *FILE*

The MPS report file to parse (default: [])

--debug

Debug mode helpful in debugging JSON parsing issues (default: False)

-v, --version

Show program's version number and exit

FILE

The MPS report file to parse (default: [])

F.88.5. Notes

The file format of the raw report files changed in HPSS version 10.1. The mps_reporter program from version 9.3 or earlier will not be able to process raw report files generated by an HPSS 10.1 system.

F.88.6. Examples

The following command will generate a report from a raw MPS report file and display the results on stdout.

```
$ mps_reporter mps_report.20220422.1
2022-04-22 10:42:05-04:00 Disk Migration for a Bitfile, <Succeeded>
    MPS Server Id                 : 15
    Thread Id                     : 0077bef4547f0000
    Storage Class                 : 1001
    Target Storage Class          : 2001
    Family ID                     : 2
    Migration Type                : Single File
    Request ID                    : 8504e4575f3b8544904332e61db43bda
    Bytes Copied                  : 1
    Bitfile ID                    : 00000007-07-0000000b-01e859f91598b5c8-14f9
2022-04-22 10:42:05-04:00 Disk Migration for a Bitfile, <Succeeded>
    MPS Server Id                 : 15
    Thread Id                     : 0017d8f4547f0000
    Storage Class                 : 1001
    Target Storage Class          : 2001
    Family ID                     : 0
    Migration Type                : Batch
    Request ID                    : 8504e4575f3b8544904332e61db43bdc
    Bytes Copied                  : 1
    Bitfile ID                    : 00000007-07-0000000b-01e859f915926790-14f9
```

```
 2022-04-22 10:42:05-04:00 Disk Migration for a Bitfile, <Succeeded>
     MPS Server Id                 : 15
     Thread Id                     : 0017d8f4547f0000
     Storage Class                 : 1001
     Target Storage Class          : 2001
     Family ID                     : 0
     Migration Type                : Batch
     Request ID                    : 8504e4575f3b8544904332e61db43bdc
     Bytes Copied                  : 1
     Bitfile ID                    : 00000007-07-0000000b-01e859f916fc0b90-14f9
 2022-04-22 10:42:05-04:00 Disk Migration Summary, <Succeeded>
     MPS Server Id                 : 15
     Thread Id                     : 00a7c9f4547f0000
     Storage Class                 : 1001
     Bytes Copied                  : 1200
     Start Time                    : 2022-04-22 10:41:22-04:00
     End Time                      : 2022-04-22 10:42:05-04:00
     Free Space at Start           : 458162176
     Free Space at End             : 458163376
     Total Space at Start          : 536805376
     Total Space at End            : 536805376
     Bitfiles Considered           : 1203
     Total Bitfiles Migrated       : 1200
         ... as Single Files       : 1
              ... in Batches       : 1199
     Number of Batches             : 26
     Number of Aborted Files       : 0
     Number of Busy Files          : 0
     Number of BFS API Errors      : 3

<...>

 2022-04-22 18:25:54-04:00 Disk Migration for a Bitfile, <<ERROR>>
     MPS Server Id                 : 15
     Thread Id                     : 00c7af4df47f0000
     Storage Class                 : 1001
     Target Storage Class          : 2001
     Family ID                     : 0
     Error Code                    : -1629
     Migration Type                : Single File
     Request ID                    : 9504e4575f3b8544904332e61db43bda
     Bytes Copied                  : 0
     Bitfile ID                    : 00000007-07-0000000b-01e85a3a4afaed3a-2de9
 2022-04-22 18:25:54-04:00 Disk Migration Summary, <Succeeded>
     MPS Server Id                 : 15
     Thread Id                     : 00e7b34df47f0000
     Storage Class                 : 1001
     Bytes Copied                  : 408944640
     Start Time                    : 2022-04-22 18:25:46-04:00
     End Time                      : 2022-04-22 18:25:54-04:00
     Free Space at Start           : 2031616
```

```
        Free Space at End             : 410976256
        Total Space at Start          : 536805376
        Total Space at End            : 536805376
        Bitfiles Considered           : 51
        Total Bitfiles Migrated       : 39
            ... as Single Files       : 39
                 ... in Batches       : 0
        Number of Batches             : 0
        Number of Aborted Files       : 1
        Number of Busy Files          : 0
        Number of BFS API Errors      : 1

    The following command  will generate a summary report from a raw
    MPS report file, via stdin, and display it on stdout.

    $mps_reporter -b -s < mps_report.20220422.1

    2022-04-22 10:36:33-04:00 Report File Version, <Succeeded>
    2022-04-22 10:36:33-04:00 MPS Startup, <Succeeded>
    2022-04-22 10:36:34-04:00 Tape File Copy Start, <Succeeded>
    2022-04-22 10:36:34-04:00 Tape File Copy Summary, <Succeeded>
    2022-04-22 10:36:34-04:00 Disk Migration Start, <Succeeded>
    2022-04-22 10:36:34-04:00 Disk Migration Summary, <Succeeded>

    The following command will generate a report for the month of April
    and save the results in the "Report.202204" file.

    $ mps_reporter mps_report.202204??.1 > Report.202204
```

F.89. mps_reporter_legacy(7)

F.89.1. Name

mps_reporter_legacy - HPSS Migration/Purge Server Report Utility

F.89.2. Syntax

mps_reporter_legacy [-b] [-c *Storage Class ID*] [-e | -E] *file_list*

mps_reporter_legacy [-b] [-c *Storage Class ID*] [-e | -E] -s

F.89.3. Description

mps_reporter_legacy is a utility which generates user readable output from raw report files created by the Migration/Purge Server (MPS).

Each file in <file_list> is processed and the output is sent to stdout.

A report contains the following types of information based on events recorded in the MPS:

When the MPS started up.

Information about the disk migration, disk purge, or tape migration of a bitfile.

Summary information of a disk migration, disk purge, or tape migration.

F.89.4. Options

-b

Generate a brief summary report only. Note that, with this option specified, the *-e* and *-E* options are ignored.

-c

Only output migration and purge entries related to the specified storage class.

-e

Only output error entries.

-E

Suppress error entries.

-s

Read from stdin.

F.89.5. Notes

The file format of the raw report files changed in HPSS version 7.4. The mps_reporter_legacy program from version 7.3 or earlier will not be able to process raw report files generated by an

HPSS 7.4 system. However, the mps_reporter_legacy program in HPSS version 7.4 is backwardly compatible and can process raw report files generated by any version of HPSS beyond 7.0.

F.89.6. Examples

The following command will generate a report from a raw MPS report file and display the results on stdout.

```
$ mps_reporter_legacy mps_report.20180517.1
05/17/2018 10:42:05 Disk Migration for a Bitfile, <Succeeded>
    MPS Server Id                : 15
    Thread Id                    : 0077bef4547f0000
    Storage Class                : 1001
    Target Storage Class         : 2001
    Family ID                    : 2
    Migration Type               : Single File
    Request ID                   : 4541515162419265585
    Bytes Copied                 : 1
    Bitfile Object ID            : 00000007-07-0000000b-01e859f91598b5c8-14f9
05/17/2018 10:42:05 Disk Migration for a Bitfile, <Succeeded>
    MPS Server Id                : 15
    Thread Id                    : 0017d8f4547f0000
    Storage Class                : 1001
    Target Storage Class         : 2001
    Family ID                    : 0
    Migration Type               : Batch
    Request ID                   : 4541515162419265588
    Bytes Copied                 : 1
    Bitfile Object ID            : 00000007-07-0000000b-01e859f915926790-14f9
05/17/2018 10:42:05 Disk Migration for a Bitfile, <Succeeded>
    MPS Server Id                : 15
    Thread Id                    : 0017d8f4547f0000
    Storage Class                : 1001
    Target Storage Class         : 2001
    Family ID                    : 0
    Migration Type               : Batch
    Request ID                   : 4541515162419265588
    Bytes Copied                 : 1
    Bitfile Object ID            : 00000007-07-0000000b-01e859f916fc0b90-14f9
05/17/2018 10:42:05 Disk Migration Summary, <Succeeded>
    MPS Server Id                : 15
    Thread Id                    : 00a7c9f4547f0000
    Storage Class                : 1001
    Bytes Copied                 : 1200
    Start Time                   : 05/17/2018 10:41:22
    End Time                     : 05/17/2018 10:42:05
    Free Space at Start          : 458162176
    Free Space at End            : 458163376
    Total Space at Start         : 536805376
    Total Space at End           : 536805376
```

```
        Bitfiles Considered          : 1203
        Total Bitfiles Migrated      : 1200
            ... as Single Files      : 1
                 ... in Batches      : 1199
        Number of Batches            : 26
        Number of Aborted Files      : 0
        Number of Busy Files         : 0
        Number of BFS API Errors     : 3

    <...>

     05/17/2018 18:25:54 Disk Migration for a Bitfile, <<ERROR>>
        MPS Server Id                : 15
        Thread Id                    : 00c7af4df47f0000
        Storage Class                : 1001
        Target Storage Class         : 2001
        Family ID                    : 0
        Error Code                   : -1629
        Migration Type               : Single File
        Request ID                   : 14647771212046664092
        Bytes Copied                 : 0
        Bitfile Object ID            : 00000007-07-0000000b-01e85a3a4afaed3a-2de9
     05/17/2018 18:25:54 Disk Migration Summary, <Succeeded>
        MPS Server Id                : 15
        Thread Id                    : 00e7b34df47f0000
        Storage Class                : 1001
        Bytes Copied                 : 408944640
        Start Time                   : 05/17/2018 18:25:46
        End Time                     : 05/17/2018 18:25:54
        Free Space at Start          : 2031616
        Free Space at End            : 410976256
        Total Space at Start         : 536805376
        Total Space at End           : 536805376
        Bitfiles Considered          : 51
        Total Bitfiles Migrated      : 39
            ... as Single Files      : 39
                 ... in Batches      : 0
        Number of Batches            : 0
        Number of Aborted Files      : 1
        Number of Busy Files         : 0
        Number of BFS API Errors     : 1

     The following command  will generate a summary report from a raw
     MPS report file, via stdin, and display it on stdout.

     $mps_reporter_legacy -b -s < mps_report.20180517.1

     05/17/2018 10:36:33 Report File Version, <Succeeded>
     05/17/2018 10:36:33 MPS Startup, <Succeeded>
     05/17/2018 10:36:34 Tape File Copy Start, <Succeeded>
     05/17/2018 10:36:34 Tape File Copy Summary, <Succeeded>
```

```
05/17/2018 10:36:34 Disk Migration Start, <Succeeded>
05/17/2018 10:36:34 Disk Migration Summary, <Succeeded>

The following command will generate a report for the month of May
and save the results in the "Report.201805" file.

$ mps_reporter_legacy mps_report.201805??.1 > Report.201805
```

F.90. multinoded(7)

F.90.1. Name

multinoded - Parallel FTP (PFTP) Multinode Daemon.

F.90.2. Synopsis



The PFTP Multinode Daemon is normally started by the xinetd daemon.

multinoded [-ir]

F.90.3. Description

The HPSS Multinode Daemon is the mechanism that allows the HPSS Parallel FTP Client to perform a parallel transfer of a file to (from) a HPSS host using multiple (remote) processors. The HPSS Multinode Daemon needs to be configured on a number of nodes. These nodes can be an individual workstation, or they can be a addressable processor, like those found on an SP2. Once the HPSS Multinode Daemons have been configured to run on a number of nodes, a list containing those nodes needs to be created and made available to every workstation that will be using the HPSS Parallel FTP Client. This done in the HPSS.conf file.

F.90.4. Options

-r

Enable root to transfer a file using the HPSS Multinode Daemon. By default root file transfers are not allowed. Allowing root to transfer a file is a security hole.

-i

Enable identification daemon (identd) authentication. The identd must be running on every client that will be using the HPSS Multinode Daemon running with this flag.

F.90.5. Configure a Node to Run the HPSS Multinode Daemon

The HPSS Multinode Daemon is started by the inetd daemon. Two files must be modified before the inetd daemon will automatically start a process: (1) /etc/services (2) /etc/inetd.conf

An entry must be added to the /etc/services file for the HPSS Multinode Daemon. The following is an example entry similar to the one that should be added to the /etc/services file:

multinoded 2513/tcp # multinode daemon



Once you have selected a port, it MUST remain constant across EVERY node.

Furthermore, an entry must be added to the /etc/inetd.conf file for the HPSS Multinode Daemon. The following are two examples of entries, similar to the one that should be added to the /etc/inetd.conf file:

multinoded stream tcp nowait root /[path]/multinoded multinoded multinoded stream tcp nowait root /[path]/multinoded multinoded -i

F.90.6. Building a Configuration File

The multinode components are specified in the HPSS.conf file. The configuration file will contain the following information:

Control Address The IP address or machine name where a multinode daemon has been configured to run.

Data Address The IP address or name to be used by the multinode daemon to move data from the mover to the multinode daemon. If a data address is not specified, the control interface will be used as the data interface.

F.90.7. Files

/etc/services

/etc/inetd.conf (AIX)

/etc/xinetd.d/multinoded (Linux)

/var/hpss/etc/HPSS.conf

F.90.8. See Also

ftp,hpss_ftp,hpss_pftpd,netrc,pftp_client,syslog,xinetd,HPSS.conf

HPSS User's Guide

HPSS Admin Guide

F.90.9. Author

Tim Preston, Sandia National Labs, Albuquerque, NM

F.91. pftp_client(7)

F.91.1. Name

pftp_client - transfers files between a local host and a remote host

F.91.2. Synopsis

pftp_client [-bcdeghilmnptvBINPST] [-w *] [-A {attempted | required | none}]
[-C] [-R -] [*host* [*port*]]

F.91.3. Description

The pftp_client command uses the File Transfer Protocol (FTP) to transfer files between a local host and a remote HPSS or non-HPSS host. The pftp_client command can perform a parallel file transfer utilizing a single local processor (parallel transfer), multiple processors (multinode transfer), or the Transfer Agent (TA).

F.91.4. Options

-b

Sets the PDATA_ONLY BufferSize. The size may be specified in the format *256KB*.

-c

Turns on the childmode. Useful for batch mode client operations.

-d

Turns on debugging. Noisy!

-e

Turns on "echo" mode. Useful for batch mode client operations. Causes every command to be echoed to the output file along with the responses from the Daemon.

-g

Turn off globbing.

-h

Specify the standard HPSS Protocol, PDATA_AND_MOVER, regardless of the setting in the HPSS.conf file. However, this will be overridden by commands in the ~/.netrc file.

-i

Disables interactive mode.

-l

Turn on verbose TA mode.

-m

Enable the use of multiple processors to complete a parallel file transfer to/from an HPSS host.

To utilize multiple processors: the multinode daemon (multinoded) must be employed; and a parallel ftp configuration file, containing a list of nodes configured to run the multinode daemon, must be available.

-n

Disables autologon mode (\~/.netrc file).

-p

Specify the WAN HPSS Protocol, PDATA_ONLY, regardless of the setting in the HPSS.conf file. However, this will be overridden by commands in the \~/.netrc file.

-t

Turns on packet tracing.

-v

Sets verbose mode. Noisy!

-w #

Sets the parallel stripe width to VAL.

-A \{*required | attempted | none}*

Specify secure authentication requirements. "required" means that the client must complete secure authentication with the server. If that fails, the client will exit. "attempted" means that the client will attempt secure authentication, but proceed regardless of the result. "none" means that secure authentication will not be attempted. For backward compatibility, GSS is still accepted and means "required", while USER_PASS means "none".

-B

Size Specify the parallelblock size. The size may be specified in the format 4MB.

-C #

Specify the default Class of Service (COS).

-I #

Specify the TA report interval.

-N #

Specify the TA message terminator.

-P

Specify the WAN HPSS Protocol, PDATA_PUSH, regardless of the setting in the HPSS.conf file. However, this will be overridden by commands in the \~/.netrc file.

-R -

Specify a Restricted Port Range.

-S #

Specify the PTRAN Open/Close Size. The size may be specified in the format 20GB.



Other factors may prevent this from operating as the customer specifies.

-T

Terminate session on error. Useful for batch processing.

F.91.5. Issuing Subcommands

See **ftp**(1).

F.91.6. Standard FTP Compatible Subcommands

!	?	$
account	append	ascii
bell	binary	bye
case	cd	cdup
close	cr	debug
delete	dir	disconnect
form	get	glob
hash	help	image
lcd	ls	macdef
mdelete	mdir	mget
mkdir	mls	mode
modtime	mput	nlist
nmap	ntrans	open
prompt	put	pwd
quit	quote	recv
rename	reset	restart
rhelp	rmdir	rstatus
runique	send	size
status	struct	sunique
system	trace	type
user	verbose	

F.91.7. Unsupported FTP Subcommands

ebcdic, proxy, sendport, tenex

F.91.8. Extended PFTP Subcommands



Acceptable minimum abbreviations in parentheses.

autoparallel (autop)

Toggle the flag to automatically substitute parallel commands for non-parallel commands.

chgrp (chg) *groupname* filename

Change group name of a remote file.

chmod (chm) *mode* filename

Change file permissions mode of a remote file.

detailTA (det)

Turn on TA verbose mode.

features (fea)

List supported features.

getprot (getp)

Determine the current protocol state.

gettuningparms (gettun) [*destination*]

Show the configuration parameters between the Client host and other hosts.

lfappend (lfa) [-l* *localfileoffset*] [-r *remotefileoffset*] [-s *sizeoftransfer*] *local-file* [*remote-file*]*

Append from a local file system file to remote file.

lfget (lfg) [-r* *remotefileoffset*] [-l *localfileoffset*] [-s *sizeoftransfer*] *remote-file* [*local-file*]*

Get a file to local system file.

lfput (lfp) [-l* *localfileoffset*] [-r *remotefileoffset*] [-s *sizeoftransfer*] *local-file* [*remote-file*]*

Send a local file system file.

mlfget (mlfg) *remote-files*

Get multiple files to local file system.

mlfput (mlfp) *local-files*

Send multiple local file system files.

mpget (mpg) *remote-files*

Get multiple files in parallel.

mpput (mpp) *local-files*

Send multiple files in parallel.

multinode (multi)

Enable the use of multiple processors to complete a parallel file transfer to (from) an HPSS host. To utilize multiple processors: the multinode daemon (multinoded) must be employed; and an HPSS configuration file (HPSS.conf), containing a list of nodes configured to run the multinode daemon, must be available.

newer (new) *remote-file* [*local-file*]

Get file if remote file is newer than local file.

pappend (papp) [-l* *localfileoffset*] [-r *remotefileoffset*] [-s *sizeoftransfer*] *local-file* [*remote-file*] *

Append a local file to a remote file in parallel.

passive (pass)

Toggle passive transfer mode. (passive mode NOT supported by HPSS PFTP Daemons for parallel transfers except for pdatapush protocol)

pdata (pdata)

Set the Parallel Transfer mode to use the PDATA_ONLY mechanism.

pdatapush (pdatap)

Set the Parallel Transfer mode to use the PDATA_PUSH mechanism.

pget (pg) [-r* *remotefileoffset*] [-l *localfileoffset*] [-s *sizeoftransfer*] *remote-file* [*local-file*]*

Retrieve one file in parallel.

pmover (pm)

Set the Parallel Transfer mode to use the PDATA_AND_MOVER mechanism.

pput (pp) [-l* *localfileoffset*] [-r *remotefileoffset*] [-s *sizeoftransfer*] *local-file* [*remote-file*]*

Send one file in parallel.

protect (prot) [*clear* | *safe* | *private*]

Set protection level for file transfer. If no level specified, reports the current level.

quote site (quo) *line-to-send*

site (sit) *line-to-send*

Send site specific command to remote server. Try "rhelp site" or "site help" for more information. Supported commands:

chgrp *groupname* filename

Change group ownership of remote file.

chmod *mode* filename

Change file permissions of remote file. Octal value between 0 an 777 may be used. The resulting permissions will be the exact value provided; e.g., "site chmod 4 myfile" will result in the permissions -------r-- on myfile. The extended mode bits setuid, setgid, etc. are NOT modified by any octal specification. To modify these bits use the symbolic format of the chmod command;

e.g., "site chmod u-s,g+srwx,o=rx myfile". This would yield the permission -rwxrwsr-x on myfile (if the rwx permissions ere already present on myfile). NOTE: the setuid and save text attribute have no meaning to HPSS. The setgid bit has no meaning to HPSS on files; but, is used for "inheritance" on directories.

chown *newowner* filename

Change ownership of remote file. (Only usable by root user)

chuid *uid* filename

Change ownership of remote file. (Only usable by root user)

idle *max-idle-time*

Set limit on time a connection can remain idle.

setcos #

Set the preferred Class of Service.

stage *filename*

Command HPSS to stage a file.

symlink *linkname* filename

Establish a symbolic link.

wait [#(>= -1) | inf | none]

Sets the wait-on-stage time. If not specified, reports the current value.

acl *subcommand*

Change/ Add ACLs to HPSS objects.

See **hacl**(7) for ACL format and available commands.

recursive *{mput* | mget | mdelete} {local-directory | remote-directory | remote-directory}

Recursively perform the following m-command on the given directory

reget (reg) *remote-file* [*local-file*]

Get file restarting at end of local file.

san3p (san3p)

Toggle SAN3P support on/off.

setpblocksize (setpb) [*size*]

Set the block size of the parallel stripe. If no size specified, reports the current block size.

setpwidth (setpw) [*width*]

Set the width of the parallel stripe. If no width specified, reports the current width.

setsockbufsize (sets) [*size*]

Set the socket buffer size. If no size specified, reports the current size.

setxferbufsize (setx) [*size*]

Set the transfer buffer size. If no size specified, reports the current size.

tareport (tar) [*#*]

Set the TA report interval (%). If no interval specified, reports the current interval.

toggleTA (tog)

Toggle use of the Transfer Agent on/off.

umask (um) [*mask*]

Set umask on remote side. If no mask specified, reports the current mask.

F.91.9. Building a Configuration File

A configuration file will be used to specify different IP addresses or remote nodes on which the data transfer processes are to be spawned by HPSS Parallel FTP Client (multinode mode) or which network interfaces to use for network striping (multihomed mode). The default configuration file is /var/hpss/etc/HPSS.conf. The user can override the path to this file with the HPSS_CFG_FILE_PATH environment variable. It is the Administrator's responsibility to correctly configure and maintain the HPSS.conf file as well as implementing the multinoded components on all remote systems. See **HPSS.conf**(7)

F.91.10. Files

~/.netrc

/var/hpss/etc/HPSS.conf

F.91.11. Environment

HPSS_CFG_FILE_PATH Alternate path to HPSS.conf

F.91.12. See Also

ftp,hacl,hpss_ftp,hpss_pftpd,multinoded,netrc,HPSS.conf

HPSS User's Guide

HPSS Admin Guide

F.91.13. Author

Tim Preston, Sandia National Labs, Albuquerque, NM

F.92. plu(7)

F.92.1. Name

plu - List files that have a purge record in the HPSS metadata.

F.92.2. Syntax

plu [-i *SUBSYS_ID*] [-S *SC_ID*] [-l | -L | -x]
[-I] [-f] [-t] [-z | -Z] [-c] [-C] [-a] [-A] [-r] [-R] [-o] [-O]
[-F] [-n] [-H] [-h] [-v] [-s *Schema*] [-g *Global*]
[-p *Principal*] [-m *Mech*] [-k *Authenticator*]

F.92.3. Description

"plu" is an HPSS System Maintenance Utility used to generate a list of full pathnames of files in a given subsystem that have purge records in the HPSS metadata. The output will be grouped by storage class with each group preceeded by a listing header. You can use the -H option to suppress the header lines. By default, the files from all storage classes are listed. You can use the -S option to limit the listing to files in a particular storage class. The list is written to STDOUT, with error and diagnostic (verbose mode) messages written to STDERR.

NOTE:

1. There are a number of options to include file attributes such as file size, file family ID, file creation time, etc in the output.
2. Within a storage class listing, the files are not listed in any particular order. For some storage classes the list of purgeable files can be very large and sorting can be expensive. If needed, you can sort the list after the fact by including the sort field of interest in the output, then use the UNIX sort utility on the resulting file.
3. By default, the listing will NOT include files that are purge locked. See the -l and -x and -L options to control the printing of purge locked files.

F.92.4. Options

-i *SUBSYS_ID*

Specify the HPSS subsystem. If not specified,the env varibale HPSS_SUBSYS_ID will be used.

-S *SC_ID*

List files from storage class <SC_ID> only. The default is to list files from all storage classes.

-I

Print the storage class ID in the output lines.

-l | -L | -x

-l will only include files with purge locks (and expired locks). -L will list ONLY files with a purge lock (or expired lock). -x will list ONLY files with an expired purge lock.

Regular purge locks apply only to the storage class at the top of hierarchy. Super purge locks apply to all storage classes in the hierarchy.

Super purge locks do not expire and so are not displayed with the -x option.

The default is to exclude purge locked files from the output.

The output will include the lock status for regular purge locks in the first column with status values:

```
' '  (blank) if the file is not regular purge locked.
```

```
'L'  if the file is regular purge locked.
```

```
'X'  if the file has an expired regular purge lock.
```

The output will include the lock status for super purge locks in the second column with status values:

```
' '  (blank) if the file is not super purge locked.
```

```
'S'  if the file is super purge locked.
```

-f

Print the file family info.

-t

Print the account ID owner of the file.

-z | -Z

-z will print the file size in bytes. -Z will print the file size in bytes with commas for readability.

-c

Print the file creation time in seconds since UNIX epoch. This is useful for external sorting.

-C

Print the file creation time in human readable format.

-a

Print latest file access time in seconds since UNIX epoch. This is useful for external sorting.

-A

Print latest file access time in human readable format.

-r

Print the purge record creation time in seconds since epoch. This is useful for external sorting.

-R

Print the purge record creation time in human readable format.

-o

Print the purge record lock time in seconds since UNIX epoch. This is useful for external sorting.

-O

Print the purge record lock time in human readable format.

-F

Prefix each pathname with the name of the fileset it it resides in.

-n

Print the file UUID rather than the file pathname. The default is to generate and print the full pathname of each file, which can be slow. Use this option if the pathnames are not needed.

-H

Do NOT print a header line before listing files in each storage class. This is useful if you are going to sort the output. The default is to print the header lines.

-s *Schema*

Name of the schema. If not specified, the env variable HPSS_MM_SCHEMA_NAME will be used.

-g *Global*

Name of the global (configuration) database. If not specified,the env varibale HPSS_GLOBAL_DB_NAME will be used.

-p *Principal*

The server principal name which this utility will run under If not specified, the env variable HPSS_PRINCIPAL_SSM will be used.

-k *Authenticator*

The keytab name which this utility will run with. If not specified, the env variable HPSS_PRIMARY_AUTHENTICATOR will be used.

-m *Auth_Mech*

The Authentication Mechanism. If not specified, the env variable HPSS_PRIMARY_AUTHN_MECH will be used.

-v

Verbose output for debugging purposes, written to STDOUT.

-h

Print the usage description for this utility.

F.92.5. Examples

The following command will print the list of purgeable files in subsystem 1.

```
$ plu -i 1

FilePath
========
/home/w/welcome/cos1test1/c1t1_00001
/home/w/welcome/cos5test2/c5t2_00000
/home/w/welcome/cos5test2/c5t2_00001
/home/w/welcome/cos5test2/c5t2_00002
/home/w/welcome/cos5test2/c5t2_00003
```

The following command will print the list of files with no header from subsystem 1, storage class 8. The storage class ID will be printed. The configuration database is specified.

```
$ plu -i 1 -S 8 -I -H -g cfg
8   /home/goodwinj/plutest/file1
8   /home/goodwinj/plutest/file2
8   /home/goodwinj/plutest/file8
```

The next example will print only purge locked files (or files with expired purge locks) in subsystem 1. The output will include the storage class ID, the file family ID and the time each file was purge locked. In addition the path names will be prefixed with the fileset name.

```
$ plu -i 1 -I -f -F -O -L

LS SC  Fam       PurgeLockTime FilePath
== === === =================== ========
L  100 0   2010-11-08 11:14:25 FilesetRoot.2685:./home/w/welcome/plutest/locked
XS 100 0   2010-10-16 09:04:23 FilesetRoot.2685:./home/w/welcome/foo/expired
X  100 0   2010-09-04 12:21:44 FilesetRoot.2685:./home/w/welcome/testfile1
```

The next example shows files in a hierarchy with these storage classes:

```
   level 0     sclass 100
   level 1     sclass 101
   level 2     sclass 102
```

File carla has an unexpired regular purge lock at level 0 of the hierarchy, storage class 100. File elwood has an expired purge lock at level 0 and a super purge lock. File abby has no regular purge lock but does have a super purge lock. File beatrice has no purge lock of either kind. These commands demonstrate that plu reports the super purge lock for all levels but the regular purge lock only for level 0:

```
$ plu -i 1 -S 100 -l

LS FilePath
== ========
   /home/george/beatrice
L  /home/george/carla
XS /home/george/elwood
 S /home/george/abby

$ plu -i 1 -S 101 -l

LS FilePath
== ========
   /home/george/beatrice
   /home/george/carla
 S /home/george/elwood
 S /home/george/abby

$ plu -i 1 -S 102 -l

LS FilePath
== ========
   /home/george/beatrice
   /home/george/carla
 S /home/george/elwood
 S /home/george/abby
```

F.93. quaid(7)

F.93.1. Name

quaid - HPSS smart recall tool

F.93.2. Syntax

quaid [*OPTIONS*]... [*INPUTS*]...

F.93.3. Description

Quaid is a utility which utilizes the HPSS batch recall functions to provide an out-of-the-box mass recall capability for HPSS. It is meant to allow users to stage many files from a variety of input sources in an organized fashion.

Quaid is engaged in four steps:

1. Scanning inputs from the sources
2. Applying filters to only stage matching files
3. Coalescing matching files from a minimal number of tape VVs into a set of batch stage requests
4. Issuing batch stage requests and providing status updates on the outcome of each stage

Based upon user inputs, a set of input sources is defined. If these are file lists, directories, or UDA queries then the intermediate results are placed as temporary files to allow for an arbitrary number of items to be processed. These input sources may themselves generate additional input sources to scan and process if recursion is enabled.

From those defined input sources, file objects are identified and their attributes read. At this point, filters specified by the user are applied to allow matches to remain as inputs to be processed, while non-matches are rejected.

Next, the remaining set of files is processed further to build a set of batch stage requests which utilizes a minimal number of tape VVs for each batch. The batches may be processed in parallel based upon user criteria.

Finally, the batches are issued. By default, the stage is periodically statused and information about each item in the active batches is posted to the report file (by default stdout). As stages complete, their status is posted to stdout.

Once all stages have processed, the tool returns to step 1 to retrieve additional inputs from the available data sources and continues on.

It is incumbent upon the user running the tool to handle stage success messages and take appropriate action to retrieve the file data from HPSS after it is staged. File data staged by quaid is subject to the same purge policy criteria as any other stage.

F.93.4. Input Methods

--file, -f *file-name*

Specify an HPSS file name to stage.

--file-list, -F *file-list-file*

Specify a file containing HPSS file names to stage.

--bitfile-list, -B *bitfile-list-file*

Specify a file containing HPSS bitfile IDs to stage.

Note that this file can contain either bitfile IDs or bitfile ID and distribution key pairs in the form BFID:KEY, for example:

00000539-07-00000003-01e93ad4493e9a66-1a39 00000539-07-00000003-01e93ad4493e9a66-1a39:1231

--xquery-match, -u *file-match-xquery*

Specify an XQuery, run against the HPSS namespace, whose matches will be staged. For example, -u *$DOC/hpss/test* will match any files with an UDA path of /hpss/test.



Any XQuery paths queried should be indexed! Talk with your HPSS administrator to set up an index.

F.93.5. Options

--auth-type, -a *auth-type*

Authentication type.

--login-principal, -p *user*

User Principal.

--keytab, -k

Indicates a keytab will be used.

--keytab-file, -t *keytab*

Keytab path.

--subsystem-id, -I *subsystem-id*

HPSS Subsystem Id (default=1).

--queue-status, -P

Obtain detailed information about stage queue status. For requests which are active, obtain detailed status about the request's position in the Core Server's tape scheduler. This will cause quaid to report the position of the request in the queue. A position of -1 indicates that the request has not been scheduled yet.

--files-per-batch, -b *batch-size*

Number of files to be processed within a batch (default=999).

--filter, -l *filter-string*

Defines a filter to be applied to the scanned objects. See the section below on filters for more information.

--process-shelf, -S

Process files on shelved tapes.

--process-trash, -T

Process files in the trashcan.

--quiet, -s

Quiet Mode - Do not display errors about skipped files.

--report-interval, -i *report-interval*

Report interval, in seconds (default=5).

--report-file, -o *report-file*

Report file (default=stdout).

--dry-run, -R

Dry Run Mode. In this mode, everything is processed normally except that the generated batches are displayed instead of processing. There is no request for data movement in dry run mode.

--recursive, -r

Recurse directories. It should be noted that each subdirectory counts as an input source. The maximum number of input sources is *4096*. If recursion results in more via directory listings, those directories will be skipped. (default=off)

--data-directory, -m

Temporary directory for temporary input source data. Input source files are opened in an unlinked state, but still consume space on the file system until they are exhausted or quaid ends.

--disable-far, -g

Explicitly disable Full Aggregate Recall for this operation. Full Aggregate Recall can dramatically improve stage performance when many files are co-located within an aggregate, however it can also cause significant amounts of unneeded data to be staged if very few files within an aggregate are needed.

--enable-far-files, -A *files-per-FAR*

Modify the threshold for disabling a FAR recall for files within a batch. If less than the specified number of files are located in the same tape section within the batch, then FAR will be disabled for those files and they will be staged individually.

To force quaid to always allow FAR when it is enabled on the COS, specify 1.

This option has no effect if FAR is disabled on the COS.

--thread-count, -n *threads*

Work off the batches in multiple threads. The default thread count is one. Adding additional threads means that additional generated batches will be worked in parallel (see the dry run output for a list of the batches). Batches are generated such that the same volumes are always grouped together in contiguous batches to limit threading's potential for increasing the requested number of VVs. There is currently a limit of 16 threads.

--scan-thread-count, -N *threads*

Scan input sources in parallel. This includes getting metadata for individual files in the input list. The default is 5. There is currently a limit of 10 threads.

--complete-callback, -C *callback-path*

Specify a callback location. In addition to reporting the stage completion to stdout, quaid will call the supplied executable after each stage finishes. It is called with the following arguments:

```
arg1: Return Code
arg2: Request ID
arg3: BitfileID:BitfileHash
arg4: Caller UID
arg5: Caller GID
arg6: File Path
```

The output will be similar to: 0 d5983acccabb11e9aff45254000db6fb 0000347d-07-00000002-01e9a26392e7f4a4-30e1:29186 0 0 /testfile

See tools/quaid/quaid_callout_example.ksh for an example callout script.



Since a process is forked for every successful stage, the resulting process should be short lived. If longer processing is required, the callout should dump the results to an intermediary location to allow for consumption by other processes.

--usage,--help,-?,-h

Display usage.

F.93.6. Note

Quaid responds to the HPSS_QUAID_MAX_FILES environment variable. This can be set either in the shell prior to invocation or in the HPSS env.conf file. This variable increases the maximum number of files that quaid will allow to be processed before it begins rejecting additional files. The default is 10 million.

F.93.7. Examples

Example - stage a directory

```
    #./quaid -f /home/hpssuser
```

Example - stage two directories and a file

```
#./quaid -f /home/hpssuser -f /home/hpssuser/project -f /home/hpssuser/file
```

Example - file list filelist format: /home/hpssuser /home/hpssuser/project

```
#./quaid -F filelist
```

Example - filter by COS (COS == 1)

```
#./quaid -F filelist -lcos:1:eq
```

Example - only stage files which match a UDA (an attribute of /hpss/color set to *red*)

```
#./quaid -u '$DOC/hpss[color=red]'
```

F.93.8. Output Format

Quaid has three output streams: a results stream, an error stream, and a status stream.

The results stream is printed to stdout and contains information such as:

```
thread 0 running (batch 0)
SUCCESS:/a/test/file2:1898578002
SUCCESS:/a/test/file3:1898578002
ERROR (28):/a/test/file4:1898578002
```

The "SUCCESS" entries are files with data on tape that are fully staged. The message format:

```
SUCCESS:<filename>:<request id>
ERROR (<error code>):<filename>:<request id>
```

The error stream prints to stderr, and contains information that is not about the result of a stage request, such as:

```
/a/test/file skipped - file is not on tape
```

The status stream prints to stdout by default, but can be changed with the -o option. The status stream prints the periodic stage status for each batch, and it looks like:

```
status path - /tmp/dtm3 - active:22495009:-1
```

The message format is one bitfile per line:

```
status path - <filename> - <active/queued/error/unknown>:<request id>:<queue position>
```



Files which have completed may show up with a status of *unknown*

A -1 queue position indicates that either the stage request is not in the tape scheduler yet, or tape scheduler status was not requested (-P option). All request statuses will contain a -1 queue position unless the -P option is requested.

F.93.9. Filters

Filters are specified in filter-name:value:operation format. For example, uid:300:eq is a filter which matches files whose uid attribute is equal to 300.

Table 39. Available filters

Name	Description
family	File Family Id (ex: 1)
cos	Class of Service Id (ex: 1)
acct	Account Id (ex: 301)
uid	UID (ex: 301)
gid	GID (ex: 301)
Subsystem	Subsystem (ex: 1)
vv	PV Name (ex: V00001)
uda	User-defined Attribute (ex: $DOC/hpss[color=*blue*]) NOTE: only eq and ne operations are available for the UDA filter. XQuery provides finer-grained matching constructs.

Table 40. Available filter operations

Name	Description
eq	Equal
ne	Not equal
gt	Greater than
ge	Greater than or equal
lt	Less than

le	Less than or equal

F.93.9.1. Limitations and Performance

Up to 20 filters may be specified. If ANY filter is matched, the file will be included in the input list (rather than having to match ALL filters).

Unlike the -u option, no index is needed for the UDA filter. However, it is still more resource-intensive than other filter types.

F.94. rc.hpss(7)

F.94.1. Name

rc.hpss - Start or stop HPSS components

F.94.2. Syntax

rc.hpss [-o] [-m] [-d] [-p] [{*start*|*stop*}]

F.94.3. Description

This utility starts or stops the HPSS Startup Daemon, and, where applicable, the SSM System Manager and prerequisite software like DB2, Kerberos, and LDAP on the current host. If no "start" or "stop" option is given, "start" is assumed. The utility attempts to detect whether various components are running, and will not restart things that are already running.

F.94.4. Options

-o

Indicates that the output of the script (and the programs it starts) should be redirected to /dev/console.

-m

Start or stop only the SSM System Manager.

-d

Start or stop only the Startup Daemon.

-p

Start or stop only the prerequisite software.

F.94.5. Examples

Start all HPSS components:

```
# rc.hpss
```

Stop only the Startup Daemon:

```
# rc.hpss -d stop
```

F.95. reclaim(7)

F.95.1. Name

reclaim - Reinitialize a tape volume

F.95.2. Syntax

reclaim [--age=*a*]
[--number-to-reclaim=*n*]
[--subsystem-id=*ss*]
[--storage-class=*sc*]
[--show-vvs]
[--global-db-name=*dn*]
[--db-schema-name=*sn*]
[--authenticator-name=*an*]
[--authentication-mech=*am*]
[--login-principal=*lp*]

reclaim [-a *Required age (in days) for reclaimed volume*]
[-n *Number of volumes to reclaim*]
[-i *Subsystem Id*]
[-c *Storage Class*]
[-S]
[-g *Global Database Name*]
[-s *Database Schema Name*]
[-k *Authenticator*]
[-m *Authentication Mechanism*]
[-p *Utility Principal Name*]

F.95.3. Description

"reclaim" selects empty tape Virtual Volumes (VVs) and reinitializes them so that they can be written again. The programs selects volumes in "EMPTY" state, which are volumes that have been written, reached EOM and have subsequently had all storage segments removed. HPSS requires that empty volumes be deleted from the system, then reintroduced as new VVs before they can be written again. This is a safety precaution that ensures that all data has been correctly deleted from the VV before it is reused. HPSS will not delete a VV that has any active data on it.

Reclaim can reclaim tapes using many combinations of the available filters. The filters are age, number to reclaim, subsystem, and storage class. At least one filter must be supplied. For example, running reclaim with the subsystem-id specified will reclaim all "EMPTY" VVs on that subsystem, regardless of storage class or age.

The user can provide command line arguments that can direct reclaim to a specific Core Server, storage class, number of VVs to reclaim, or required "EMPTY" time of VVs to be reclaimed. The user cannot specify which VVs are reclaimed. The VVs which have been empty the longest are reclaimed first. Only tape VVs can be reclaimed.

In cases where no storage class is specified, it still holds true that the VVs which have been empty the longest across all storage classes will be reclaimed first. When multiple subsystems are being reclaimed, the subsystems will be processed in one by one, with each counting towards the overall number of VVs to reclaim.

When reclaiming multiple subsystems, reclaim must have access to each subsys database on the system where it is run as well as each Core Server. If the databases are not accessible on the system where reclaim is run reclaim cannot process that subsystem and will fail.

This program is commonly used after repacking tape volumes. When repack is successful, it leaves tape volumes empty, ready to be reclaimed. It may be useful to time reclaim operations to occur after scheduled database backups - in the event of a disaster which requires a restore to a point before a certain tape was repacked, this will ensure that the EMPTY tape has not been overwritten and the restore point metadata can recover the data on that tape.

VVs are reclaimed by the Core Server in a single database transaction that deletes the old VV's storage map and metadata, refreshes the PV metadata to "new PV" state, then creates a new VVID, and new VV and storage map metadata records. If any part of this process fails, the transaction fails and no change is made to any of the associated metadata. If the reclaim process succeeds, the reclaimed VV appears in the system metadata as if it had just been created, and is immediately available for use.

The user may supply the name of the global database and the database schema that contains the configuration information for the HPSS system, otherwise default values will be used.

Values for the global database name, schema name, authentication mechanism, authenticator and principal name will be taken from the user's environment if the corresponding arguments are not supplied on the command line. If these environment variables do not appear in the user's environment, default values will be used. The command line arguments override the environment.

This program may also be run via an SSM screen associated with tape storage classes with a limited feature set; SSM provides all the command line arguments.

F.95.4. Options

--age, -a *Age*

The minimum number of days the tape has been "EMPTY" in order for it to be eligible for reclaim. The default value is 0, (no age requirement).

--storage-class, -c *Storage Class*

The storage class of the volumes to select. The storage classes supported can be found from SSM through: Monitor, Storage Class List screens. This argument is not required on the command line; the default value is 0 (all storage classes).

--subsystem-id, -i *Subsystem ID*

The id of the subsystem in which the volumes reside. This is a non-zero positive number. This argument is not required on the command line and the default value is 0 (all subsystems).

--number-to-reclaim, -n *Number of VVs*

The number of tape volumes to reclaim. VVs are selected from the given storage class and reclaimed until this number is reached, or all empty VVs in the given storage class have been selected. The default is 0 (all eligible VVs).

--global-db-name, -g *Global DB Name*

The ASCII name of the global database where the basic configuration information for HPSS is found. If not specified in the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--db-schema-name, -s *Schema Name*

The ASCII name of the Db2 schema that contains the global database (-g). If not specified in the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--authenticator-name, -k *Authenticator*

The authenticator which is used to authenticate the *reclaim* utility. Reclaim is usually run as SSM. The Keytab name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--authentication-mech, -m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--login-principal-lp, -p *Principal Name*

The server principal name which this utility will run under. Reclaim is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided in the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--show-vvs, -S

Show the names of physical volumes in the selected tape volumes, but don't reclaim them. This option allows the program to perform all of the volume selection steps without making a permanent change.

F.95.5. Examples

The following command reclaims up to 10 empty tape volumes in storage class 7, found in subsystem 1. All of the remaining command line arguments are defaulted to the enviroment values, or the system defaults (this goes for all subsequent examples too).

```
$ reclaim --storage-class=7 --subsystem-id=1 --number-to-reclaim=10
```

The following command reclaims all empty tape volumes in storage class 7, found in subsystem 1, which have been empty at least 5 days.

```
$ reclaim --storage-class=7 --subsystem-id=1 --age 5
```

The following command reclaims all empty tape volumes in all tape storage classes in subsystem 1, which have been empty at least 5 days.

```
$ reclaim --subsystem-id=1 --age 5
```

The following command reclaims all empty tape volumes in all tape storage classes in all subsystems which have been empty for at least 5 days.

```
$ reclaim --age 5
```

The following command reclaims all empty tape volumes in all tape storage classes in all subsystems regardless of age.

```
$ reclaim --subsystem-id=0
```

F.95.6. See Also

repack

F.96. recover(7)

F.96.1. Name

recover - Recover data from alternate copies

F.96.2. Syntax

F.96.2.1. Command line form 1:

Display usage.

recover --help

recover -h

F.96.2.2. Command line form 2:

Display the list of volumes currently undergoing recovery or cleanup.

recover --list
[--subsys-id=*ss*]
[--global-db-name=*dn*]
[--authenticator-name=*an*]
[--authentication-mechanism=*am*]
[--login-principal=*lp*]
[--db-schema-name=*sn*]
[--debug]

recover -l
[-i *ss*]
[-g *dn*]
[-k *an*]
[-m *am*]
[-p *lp*]
[-s *sn*]
[-d]

F.96.2.3. Command line form 3:

Display summary status of one volume currently undergoing recovery or cleanup.

recover --volume-name=*pv*
--status
[--subsys-id=*ss*]
[--global-db-name=*dn*]
[--authenticator-name=*an*]
[--authentication-mechanism=*am*]
[--login-principal=*lp*]

[--db-schema-name=*sn*]
[--debug]

recover -v *pv*
-t
[-i *ss*]
[-g *dn*]
[-k *an*]
[-m *am*]
[-p *lp*]
[-s *sn*]
[-d]

F.96.2.4. Command line form 4:

Run in recovery mode (default) or cleanup mode (--cleanup-data option).

recover --volume-name=*pv*
[--subsys-id=*ss*]
[--global-db-name=*dn*]
[--authenticator-name=*an*]
[--authentication-mechanism=*am*]
[--login-principal=*lp*]
[--db-schema-name=*sn*]
[--cleanup-data]
[--debug]
[--retry-failures]
[--retry-incompletes]
[--threads=*nt*]
[--watch-interval=*ns*]

recover -v *pv*-
[-i _ss]
[-g *dn*]
[-k *an*]
[-m *am*]
[-p *lp*]
[-s *sn*]
[-c]
[-d]
[-R]
[-r]
[-T *nt*]
[-w *ns*]

F.96.2.5. Command line form 5:

Cancel the recover or cleanup (--cancel option).

recover --volume-name=*pv*
--cancel
[--subsys-id=*ss*]
[--global-db-name=*dn*]
[--authenticator-name=*an*]
[--authentication-mechanism=*am*]
[--login-principal=*lp*]
[--db-schema-name=*sn*]
[--debug]

recover -v *pv*
-C
[-i *ss*]
[-g *dn*]
[-k *an*]
[-m *am*]
[-p *lp*]
[-s *sn*]
[-d]

F.96.3. Description

The recover utility is used to recover data from an alternate copy when the data on a disk or tape volume is no longer accessible due to a media failure. When used in this way, recover is defined to be in recovery mode, which is the default mode. If no alternate copy is available, the recover utility can be used to clean up the metadata for the volume so it can be removed from the HPSS system. When used in this way, recover is said to be in cleanup mode, which is entered by specifying the --cleanup-data option.

In cleanup mode, the recover utility deletes the data from the specified volume rather than recovering from other levels in the hierarchy. This mode is useful for files for which there are no alternate copies.

In recovery mode, the recover utility reads the data for each file on the damaged volume from an alternate copy at a different level of the hierarchy. Recover stages the files to level 0 of the hierarchy, and uses "force migration" to put a new copy at level 1 of the hierarchy. Data which is force migrated will be migrated down to each level of the hierarchy regardless of whether it is out of date. In the process, the data will be written to a good volume at the damaged level of the hierarchy.

The force migration from level 0 to level 1 is started and monitored by the recover utility. Migrations from level 1 down the hierarchy are started by the MPS independently from the recover tool.

The recover utility reports summary status messages to standard error on every progress watch cycle. The default interval for the progress watch cycle is 30 seconds. It may be changed with the --watch-interval option. See the SUMMARY STATUS section of this man page for an explanation of the contents of the summary status messages.

Detailed status and error messages for each individual file are logged to a report file each progress watch cycle; see the REPORT FILE sections of this man page. This file reports the reason for any failed operations. This information can be used to rectify a problem before rerunning the recover. For example, if the reason for a failure is "No space left on device," additional space can be added to the problem storage class.

If the recover utility reports failures, it may be restarted with the --retry-failures option to retry the failed files. Before the recover is retried, the recover report file and the HPSS system log should be examined to determine the cause of the failures and the underlying problem should be addressed.

The HPSS system must be running in order for recover to work. In particular, the Core Server and MPS must be up. If the Core Server or MPS is restarted while a recover operation is underway, or if the recover utility reports that the MPS is no longer working the recover, then the recover utility must be terminated and then restarted with the --retry-incompletes option. The recover utility will then resubmit any outstanding jobs to the Core Server and/or MPS.

In the case where the recover utility reports that the MPS is no longer working on the recover, then the recover report file and the HPSS system log should be examined to determine the cause of the failures and the underlying problem should be addressed before restarting the restarting the recover utility with --retry-incompletes.

If recover is first invoked in recovery mode, it may later be rerun in cleanup mode. In this case, the data for any files which have failed recovery or for which recovery has neither been attempted nor completed will be removed from the volume. This enables the volume to be removed from the system via normal resource deletion and export operations. Data for any files which were already successfully recovered will not be removed from their new volumes. Before the recover is switched to cleanup mode, failures should be investigated and retried if possible.

Once recover has been run in cleanup mode for a volume, it cannot be rerun in recovery mode for that volume. In this case, if recover is stopped and restarted, it will continue to run in cleanup mode, whether or not the --cleanup-data option is specified on the restart.

If the recover utility is terminated before it is finished, it may be restarted and will pick up where it left off. This is true even if HPSS is restarted or the host machine is rebooted.

Even after the data is restored or cleaned up for every file on the volume, the operation is not fully complete until the recover utility has verified it and marked it as complete. The recover summary status and report file can each be used to determine whether the operation is complete. Upon completion, they will each report

Recovery of volume XXXXXX00 complete.

for recover operations or

Cleanup of volume XXXXXX00 complete.

for cleanup operations.

To cancel a recover or cleanup operation, use the --cancel (-C) option. The recover utility will cancel the request and abandon work on all files on the volume. Files which have already been completed

will not be undone. Some files which are not yet complete may actually be restored eventually. For example, if a file has been staged and then force migrated to the direct migration targets of level 0, normal migration should eventually complete all the remaining copies needed.



When a recover is canceled, all checkpoints related to partially done operations are removed. Reattempting recover on a volume after cancellation will restart the recover process from the beginning.



If recover managed to purge a file from the volume and the checkpoint is removed subsequent calls to recover would not stage, and force migrate that file, since it is no longer on the volume being recovered.

Once the recovery or cleanup operation is complete or cancelled, the damaged volume may be removed from the system by means of normal "delete resources" and "export" operations. HPSS will not allow the "delete resources" operation until the recovery or cleanup operation is complete or cancelled. Even if the recover utility is killed, HPSS will recognize when the recover or cleanup operation is not complete or cancelled and will not allow the "delete resources."



If recover is run in cleanup mode, and there is more than one level of the hierarchy, an outdated copy of the file can be left at one or more of these levels. For example, suppose a hierarchy has two levels, and data for a file was migrated to the second level, and then the file was modified but not re-migrated to the second level. If the volume at the top level of the hierarchy is cleaned up, then the out of date copy of the file remains at the second level, and will be returned to the user on a read. Before running recover in cleanup mode, consult your support representative.

F.96.4. Options

-C, --cancel

Requests that the recover or cleanup operation on the volume be cancelled.

-c, --cleanup-data

Requests that all data on the volume be removed. This option is typically used when a volume is damaged and there is no valid data at any other level of the hierarchy from which the file can be recovered. The metadata for the data must be removed so that the volume can be removed from the HPSS system.

-d, --debug

Specifies that debugging information should be displayed.

-g, --global-db-name=*Global DB Name*

Specifies the ASCII name of the global database where the basic configuration information for HPSS is found. If the global database name is not provided in the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used. The default is "cfg."

-h, --help

Specifies that usage help should be displayed.

-i, --subsystem-id=*Subsystem ID*

Specifies the ID of the subsystem in which the volume resides. If the subsystem ID is not provided in the command line, the value of the environment variable HPSS_SUBSYS_ID will be used.

-k, --authenticator-name=*Authenticator*

Specifies the authenticator with which this utility will run. If the authenticator name is not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used. The default is "/var/hpss/etc/hpss.keytab." Systems using unix authentication should use "/var/hpss/etc/hpss.unix.keytab."

-l, --list

Requests that a list of all volumes in the specified subsystem which are currently undergoing recovery or cleanup be displayed.

-m, --authentication-mech=*Mechanism*

Specifies the security mechanism used to authenticate the client. If the security mechanism is not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used. Allowed values are "unix" or "krb5."

-p, --login-principal=*Principal Name*

Specifies the server principal name which this utility will run under. If the principal name is not provided in the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used. The default is "hpssssm."

-R, --retry-failures

Requests that failed recovery or cleanup operations be retried. Operations that fail will be reported in the report file.

This option does not retry operations which succeeded. It does not restart operations on files for which recovery or cleanup are still underway, have not yet completed, and have not yet failed.

-r, --retry-incompletes

Requests that incomplete recovery or cleanup operations be retried. Operations on files are incomplete if the Core Server and/or the MPS were terminated before the operation could be completed for all files. In recovery mode, incomplete files might never have been staged or might have been staged but never migrated. In cleanup mode, the cleanup might never have been attempted.

This option does not retry operations which were actually attempted but failed.

-s, --db-schema-name=*Schema Name*

Specifies the ASCII name of the Db2 schema that contains the global database (-g). If the schema name is not provided in the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used. The default is "HPSS."

-T, --threads=*nt*

Specifies the numbers of threads used for the analysis phase, in which the recover program determines the information required for the recovery or cleanup of each file. The default number of threads is 4. The minimum allowed is 1 and the maximum allowed is 1000. Testing on some configurations showed the optimum performance for a volume with 500,000 files was achieved at 40 threads. This number is likely to vary widely depending on the environment.

-t, --status

Requests that recover run in status-only mode. The summary status of the recover or cleanup operation on the specified volume is reported. No recover operations are conducted when this option is used. It is used to check the progress of a recover operation that is already running. The --subsystem-id and --volume-name options are required with the --status option. No other options are allowed with the --status option.

-v, --volume-name=*pv*

Specifies the name of the damaged volume. The format of the name is the six character volume name followed by two zeros. The --volume-name option is required unless the --list or --help option is specified.

-w, --watch-interval=*ns*

Specifies, in seconds, the interval for the progress watch cycle. The default is 30 seconds. On each progress watch cycle, recover reports its summary status to standard error and, if necessary, updates the report file with detailed information.

The --retry-failures, --retry-incompletes, --cleanup-data, and --cancel options are mutually exclusive. Only one of these options may be specified for any single invocation of recover.

F.96.5. Alternate Copies

An alternate copy of a file is a complete and up to date copy of the file at a level of the hierarchy. A complete copy is one which includes the entire file. An up to date copy is one which is at the latest revision of the file.

The recover utility recovers damaged volumes only if an alternate copy of the data exists at another level of the hierarchy. The hierarchy must be configured in advance to support alternate copies. At the time of the recover operation, the alternate copies must have been populated with valid data by normal migration processes.

A permanent copy is one which is not configured to be purged. The recover utility may not be able to recover damaged volumes in all circumstances if the hierarchy contains only one permanent copy.

For example, a hierarchy with disk at level 0 which migrates to tape at level 1 and which purges the data at level 0 once migration is complete has two alternate copies of the file until the file is purged off level 0. But it has only one permanent copy of the data. After normal purging from level 0, only one complete and permanent copy of each file is guaranteed to exist in the hierarchy, the copy at level 1. If the volume at level 1 is damaged, it can be recovered from level 0 only if level 0 has not yet been purged.

A hierarchy with disk at level 0 which migrates to tape at both levels 1 and 2 and which does not allow purging from the tape levels has three alternate copies, two of which are permanent copies. After normal migration to both tape levels and before level 0 is purged, each level is an alternate copy. After level 0 is purged, two complete and permanent copies of the data remain, the copies at levels 1 and 2. If a tape volume at either level is damaged, it can be recovered from the copy at the other level. The damaged tape can alternately be recovered from the copy at level 0, but only if level 0 has not yet been purged.

Sites who need to be able to recover from alternate copies should ensure that they configure their hierarchies with at least two permanent copies.

F.96.6. Summary Status

The summary status is reported by recover to the standard error stream on every progress watch cycle. It includes the statistics for how many files are at each stage of the recovery or cleanup operation. For example:

2014.1125.1815 status for recovery of volume IT201700

```
        Report File:       /var/hpss/tmp/recover.IT201700.2014.1125.181500.report.txt
        Request submitted: 2014.1125.1815
        Operational mode:  RECOVERY: Attempting to recover data
        Operation status:  RECOVERY IN PROGRESS
              2 total files
              0 files busy and cannot be recovered or cleaned up now:
                  0 being recovered or cleaned up for a different volume
                  0 being force migrated by another application
              0 files to be cleaned up instead of recovered:
                  0 cleaned up successfully
                  0 could not complete pre-cleanup analysis
                  0 cleanup failed
                  0 were deleted before they could be cleaned up
                  0 still awaiting cleanup attempt
              2 files to be recovered:
                  0 recovered successfully
                  0 not eligible for recovery: no alternate copy
                  0 could not complete pre-recover analysis
                  0 pre-recover purge failed
                  0 stage to level 0 failed
                  0 migrate from level 0 failed
                  0 were deleted before they could be recovered
                  2 still awaiting recovery attempt:
                      0 awaiting pre-recover analysis
                      2 awaiting stage to level 0
                      0 staged to level 0, awaiting migration from level 0
                      0 migrated from level 0, awaiting final verification
              0 retries have been conducted for files on this volume
```

The operational mode may be RECOVERY, CLEANUP, SWITCHING TO CLEANUP, CANCEL, or NONE.

In SWITCHING_TO_CLEANUP mode, recover was initially run in recover mode and then re-run in CLEANUP mode, and it is still in the process of changing each file to run in CLEANUP mode. A mode of CANCEL means the recover or cleanup operation is in the process of being cancelled. A mode of NONE means recover has not yet been run on the volume or it is in the very early stages of initialization. It is rare to see a mode of NONE unless recover is run with the "--status" option on a volume for which a recover has not been submitted.

The operation status will be one of:

NONE

Initialization has not yet begun. This is a very short stage and it is rare to see a recover operation in it.

INITIALIZING

Recover is initializing.

RECOVERY IN PROGRESS

A recovery operation is underway.

CLEANUP IN PROGRESS

A cleanup operation is underway.

COMPLETE

The recovery or cleanup operation is complete.

CANCEL IN PROGRESS

The recovery or cleanup operation is in the process of being cancelled.

CANCEL COMPLETE

The recovery or cleanup operation has been cancelled.

The categories of files in the summary status report are:

total files

The total number of files recover found on the volume.

files busy and cannot be recovered or cleaned up now

The total number of files which are in use by some other recover or related operation. The current recover operation will ignore these until it completes the "non-busy" files it found on the volume.

This category has these subcategories:

being recovered or cleaned up for a different volume

A recover process for a different volume is processing the file. This could happen if recover were run for two volumes at different levels of the same hierarchy.

being force migrated by another application

A related application is processing the file.

file to be cleaned up instead of recovered

Files which are to be removed from the volume instead of recovered. If the --cleanup-data option was specified with the first invocation of recover for this volume, this will match the "total files" less any busy files. If recover was run first in recovery mode and later in cleanup mode, and some files were successfully recovered before we entered cleanup mode, this number will be less than "total files."

This category has these subcategories:

cleaned up successfully

These files are finished and cleaned up.

could not complete pre-cleanup analysis

Recover could not determine the necessary information about the file to begin the cleanup operation.

cleanup failed

The cleanup failed for these files.

were deleted before they could be cleaned up

These files were deleted after the cleanup process began.

still awaiting cleanup attempt

The cleanup process has not yet begun for these files.

file to be recovered

Files which are to be recovered from the volume. If recover was never run in cleanup mode, this will match the "total files" less the busy files. If recover was run first in recovery mode and then later in cleanup mode, and some but not all files had already been recovered, it will represent the number of files recovered before the switch to cleanup mode.

This category has these subcategories:

recovered successfully

These files are finished and recovered.

not eligible for recovery: no alternate copy

There is no alternate copy for these files. They cannot be recovered using the recover utility.

could not complete pre-recover analysis

Recover could not determine the necessary information about the file to begin the recovery operation.

pre-recover purge failed

In some cases, recover must purge the file off the damaged volume before recovery can proceed. These are the files for which that purge failed.

stage to level 0 failed

These files could not be staged to level 0.

migrate from level 0 failed

These files could not be migrated from level 0.

were deleted before they could be recovered

These files were deleted after the recovery process began.

still awaiting recovery attempt

These are the files which haven't yet succeeded or failed recovery.

This subcategory has these subcategories of its own:

awaiting pre-recover analysis

These files are waiting for recover to determine the necessary information about them to proceed with recovery.

awaiting stage to level 0

These files are awaiting stage to level 0.

staged to level 0, awaiting migration from level 0

These files have been staged to level 0 and are awaiting migration from level 0.

migrated from level 0, awaiting final verification

These files have been migrated from level 0. They are awaiting migration to any further levels in the hierarchy, if necessary, and final confirmation by recover that they are complete.

retries

This is the number of retries which have been conducted for the volume. One retry is counted for every time any file is retried.

F.96.7. Report File

The report file includes a line for each file at most stages of the recovery or cleanup operation. For failed operations on files, it includes the reason for the failure.

The report file is updated, if necessary, on each progress watch cycle of recover. If no file status has changed, the report file is not updated. Some stages may not be reported if the file passed through two or more stages between progress watch cycles.

The report file also includes status information from the startup, restart (if any), and final verification stages of recover.

Rudimentary performance monitoring information is written to the report file for each phase of the recover or cleanup operation. There are three phases and the third phase has 2 parts:

1. Analysis Phase

2. Progress Monitoring Phase
3. Final Verification Phase Part 1
4. Final Verification Phase Part 2

Normally the Analysis and Final Verification performance information is written only once, at the end of the associated phase, but it can be written multiple times if recover is retried. The Progress Monitoring information is written once for every progress monitoring cycle.

Example report file:

```
2016.0819.11:12:16 Starting recover program for volume IT100400

2016.0819.11:12:16 Analyzing damaged volume....
2016.0819.11:12:16 FsetRt: ./TM.0052: Beginning analysis
2016.0819.11:12:16 FsetRt: ./TM.0052: 3145728 bytes purged from level 0
2016.0819.11:12:16 FsetRt: ./TM.0052: submitted for recover
2016.0819.11:12:16 FsetRt: ./TM.0035: Beginning analysis
2016.0819.11:12:16 FsetRt: ./TM.0035: 3145728 bytes purged from level 0
2016.0819.11:12:16 FsetRt: ./TM.0035: submitted for recover
2016.0819.11:12:16 FsetRt: ./TM.0071: Beginning analysis
2016.0819.11:12:16 FsetRt: ./TM.0071: 3145728 bytes purged from level 0
2016.0819.11:12:16 FsetRt: ./TM.0071: submitted for recover
<....>
2016.0819.11:12:21 FsetRt: ./TR.0344: Beginning analysis
2016.0819.11:12:21 FsetRt: ./TR.0344: 3145728 bytes purged from level 0
2016.0819.11:12:21 FsetRt: ./TR.0344: submitted for recover
2016.0819.11:12:21 FsetRt: ./TM.0028: Beginning analysis
2016.0819.11:12:21 FsetRt: ./TM.0028: 3145728 bytes purged from level 0
2016.0819.11:12:21 FsetRt: ./TM.0028: submitted for recover

2016.0819.11:12:21 PERFORMANCE RATE: Analysis phase:
                   47.400 files per second analyzed.

2016.0819.11:12:21 PERFORMANCE RATE: Progress Monitoring phase:
                   237 files in 0 seconds examined.
2016.0819.11:12:22 FsetRt: ./TM.0052: recovery of file complete
2016.0819.11:12:22 FsetRt: ./TM.0035: recovery of file complete
2016.0819.11:12:22 FsetRt: ./TM.0071: recovery of file complete
<...>
2016.0819.11:12:37 FsetRt: ./TR.0281: recovery of file complete
2016.0819.11:12:37 FsetRt: ./TR.0066: recovery of file complete
2016.0819.11:12:37 FsetRt: ./TR.0003: recovery of file complete

2016.0819.11:12:52 PERFORMANCE RATE: Progress Monitoring phase:
                   237.000 files per second examined.
2016.0819.11:12:52 FsetRt: ./TR.0211: recovery of file complete
2016.0819.11:12:52 FsetRt: ./TR.0188: recovery of file complete
2016.0819.11:12:52 FsetRt: ./TR.0080: recovery of file complete
<...>
```

```
    2016.0819.11:13:07 FsetRt: ./TM.0074: recovery of file complete
    2016.0819.11:13:07 FsetRt: ./TR.0344: recovery of file complete
    2016.0819.11:13:07 FsetRt: ./TM.0028: recovery of file complete

    2016.0819.11:13:23 PERFORMANCE RATE: Progress Monitoring phase:
                       237.000 files per second examined.

    2016.0819.11:13:53 PERFORMANCE RATE: Progress Monitoring phase:
                       237 files in 0 seconds examined.

    2016.0819.11:13:53 Verifying that volume is complete.  Current status:

    2016.0819.11:13:53 status for recovery of volume IT100400:
        Report File:
/var/hpss.iris/tmp/recover.IT100400.2016.0819.111215.report.txt
        Request submitted: 2016.0819.11:12:15
        Operational mode:  RECOVERY: Attempting to recover data
        Operation status:  RECOVERY IN PROGRESS

            237 total files
              0 files busy and cannot be recovered or cleaned up now:
                  0 being recovered or cleaned up for a different volume
                  0 being force migrated by another application
              0 files to be cleaned up instead of recovered:
                  0 cleaned up successfully
                  0 could not complete pre-cleanup analysis
                  0 cleanup failed
                  0 were deleted before they could be cleaned up
                  0 still awaiting cleanup attempt
            237 files to be recovered:
                237 recovered successfully
                  0 not eligible for recovery: no alternate copy
                  0 could not complete pre-recover analysis
                  0 pre-recover purge failed
                  0 stage to level 0 failed
                  0 migrate from level 0 failed
                  0 were deleted before they could be recovered
                  0 still awaiting recovery attempt:
                      0 awaiting pre-recover analysis
                      0 awaiting stage to level 0
                      0 staged to level 0, awaiting migration from level 0
                      0 migrated from level 0, awaiting final verification
              0 retries have been conducted for files on this volume


    2016.0819.11:13:55 237 entries deleted from force migrate table.
                       0 files still on volume.

    2016.0819.11:13:55 PERFORMANCE RATE: Final verification phase part 1:
                       0 seconds to verify 0 files still on volume.
```

```
    2016.0819.11:13:55 PERFORMANCE RATE: Final verification phase part 2:
                       118.500 files per second processed from force migrate table.
    2016.0819.11:13:55 Recovery of volume IT100400 complete.
```

F.96.8. Examples

F.96.8.1. Run recover in recovery mode

Command:

```
            recover -i 1 -v IT102100
```

This says to run recover on volume IT102100 in subsystem 1. In this example, we have 305 files on the volume.

Recover will first issue messages to indicate it is initializing itself and analyzing the volume:

```
        Connected to core server.
        Connected to MPS.
        Analyzing damaged volume....
        100 files analyzed...
        200 files analyzed...
        300 files analyzed...
        Analysis complete.  305 total files found on damaged volume.
```

Once analysis is complete, recover will begin reporting the summary status on every progress watch cycle:

```
            Checking recover progress....

            2014.1125.1815 status for recovery of volume IT201700:
                Report File:
/var/hpss/tmp/recover.IT201700.2014.1125.181500.report.txt
                Request submitted: 2014.1125.1815
                Operational mode:  RECOVERY: Attempting to recover data
                Operation status:  RECOVERY IN PROGRESS
                305 total files
                  5 files busy and cannot be recovered or cleaned up now:
                      5 being recovered for a different volume
                      0 being force migrated by another application
                  0 files to be cleaned up instead of recovered:
                      0 cleaned up successfully
                      0 could not complete pre-cleanup analysis
                      0 cleanup failed
                      0 were deleted before they could be cleaned up
                      0 still awaiting cleanup attempt
                300 files to be recovered:
```

```
                     10 recovered successfully
                      0 not eligible for recovery: no alternate copy
                      0 could not complete pre-recover analysis
                      0 pre-recover purge failed
                      0 stage to level 0 failed
                      0 migrate from level 0 failed
                      0 were deleted before they could be cleaned up
                    290 files still awaiting recovery attempt:
                          0 awaiting pre-recover analysis
                         80 awaiting stage to level 0
                        110 staged to level 0, awaiting migration from level 0
                        100 migrated from level 0, awaiting final verification
                  0 retries have been conducted for files on this volume

             Waiting 30 seconds before next progress check....
```

Once each file reaches the complete state, recover enters its final verification stage. It reanalyzes the volume to make certain that there are no files remaining. The output will be something like:

```
             Verifying that volume is complete.....
             2014.1125.2005 Recovery of volume IT201700 complete.
             See detailed reports in
/var/hpss/tmp/recover.IT201700.2014.1125.181500.report.txt
```

F.96.8.2. Run recover in cleanup mode

Command:

```
             recover -i 1 -v X0202200 --cleanup-data
```

This says to run recover in cleanup mode on volume X0202200 in subsystem 1.

F.96.8.3. List volumes currently undergoing recovery or cleanup

Command:

```
             recover -i 1 --list
```

This says to list any volume in subsystem 1 that is undergoing recovery or cleanup.

Example output:

```
             Volumes currently undergoing recovery:

             VVID:
             x'129C6501F6EBE311862A0019B9C3B883E82FC2E06833DE119BAE000000000101'
```

```
            PV001:  IT201700

            VVID:
            x'5C9D6501F6EBE311862A0019B9C3B883E82FC2E06833DE119BAE000000000101'
            PV001:  IT20200
```

F.96.8.4. Display the status of a volume currently undergoing recovery or cleanup.

Command:

```
            recover -i 1 --status -v IT202000
```

This says to display the recovery or cleanup status of volume IT202000 which is in subsystem 1.

Example output:

```
            Checking recover progress....

            2014.1125.1815 status for recovery of volume IT202000:
            Report File:
/var/hpss/tmp/recover.IT202000.2014.1125.181500.report.txt
            Request submitted: 2014.1125.1815
            Operational mode:  RECOVERY: Attempting to recover data
            Operation status:  RECOVERY IN PROGRESS
                4 total files
                2 files busy and cannot be recovered or cleaned up now:
                    2 being recovered for a different volume
                    0 being force migrated by another application
                0 files to be cleaned up instead of recovered:
                    0 cleaned up successfully
                    0 could not complete pre-cleanup analysis
                    0 cleanup failed
                    0 were deleted before they could be cleaned up
                    0 still awaiting cleanup attempt
                2 files to be recovered:
                    0 recovered successfully
                    0 not eligible for recovery: no alternate copy
                    0 could not complete pre-recover analysis
                    0 pre-recover purge failed
                    0 stage to level 0 failed
                    0 migrate from level 0 failed
                    0 were deleted before they could be recovered
                    2 files still awaiting recovery attempt:
                        0 awaiting pre-recover analysis
                        0 awaiting stage to level 0
                        0 staged to level 0, awaiting migration from level 0
                        2 migrated from level 0, awaiting final verification
                0 retries have been conducted for files on this volume
```

F.96.9. Notes

The recover utility must use the principal of an authorized user. The best way to accomplish this is to take the default values for the principal, which is hpssssm, and for the authenticator, which is the HPSS system keytab. See the details for the --login-principal (-p) and --authenticator-name (-k) options in the OPTIONS section of this man page. The utility should in this case be executed as root since the HPSS system keytab is readable only by root.

The VV Condition of a volume being recovered must be set to DOWN before the recover utility is invoked and remain DOWN until the recovery or cleanup operation is complete.

Except in status-only mode, only one recover operation per volume may be run at the same time. Recover may be run in status-only mode by specifying the --status option. If a recover process is already running for a volume and a second recover is issued without the --status option, the second process will enter status-only mode anyway.

If any level of a hierarchy is disk which migrates to tape, recovered files written to those tape levels will be aggregated if the migration policy of the disk storage class specifies aggregation. If the disk migration policy is not configured for aggregation, the files recovered to those tape levels will not be aggregated on the new volume. Recover does not aggregate the recovered files if the tape volume is at level 0 or if data is migrated to the volume from tape.

Since files with data on the damaged volume are first staged to the top level of the hierarchy, the storage class at that level must have enough room to hold all of these files and still support regular ingest. If it does not, both recover and regular ingest of files can fail. Failed recover operations can be retried with the --retry-failures options once enough space is made available.

Recover stage requests which fail due to a busy system will be retried up to two times by the MPS.

If the purge action isn't disabled or suspended, a recover stage request that fails because the disk storage class has run out of free space will start a purge of that disk storage class. The recover tool will retry the failed stage requests up to two times. Purge will be run within the rules of the purge policy for the disk storage class that is out of space. Thus recover may cause fully-migrated files to be purged from disk storage classes in order to make room for files that are being recovered. Purge policy adjustments and/or adding disks may still be required.

If the top level of the hierarchy is disk, files staged there by the recover program will be purge locked until they are migrated from this level. The purge locks will be removed when the files have been migrated from the top level.

All files undergoing recovery or cleanup will be super purge locked by the recover program. The super purge lock is a means of insuring that the last copy of the file data cannot be accidentally removed. It prevents the purge of any level of the file, disk or tape, except by force. In recovery mode, the recover program will attempt to remove the super purge lock for each file when that file has been migrated from the top level. In cleanup mode, the recover program will attempt to remove all of the super purge locks when the cleanup of all files has been completed. Unless the last copy of the file was intentionally removed by running recover in cleanup mode, the super purge lock will not be removed for any file unless the file still retains a safe copy.

The MPS does not make provision for recover operations in its migration stream counts. An

adequate number of tape drives beyond those configured for normal migration streams must be available to support the recover operation.

The recover utility should not be executed on multiple volumes in the same hierarchy at the same time, as the multiple processes will attempt to use the same resources, resulting in inefficient usage of the system.

The definition of $HPSS_PATH_TMP should not be changed until the recover process is complete. The recover utility should not be executed for the same damaged volume from two separate machines, even sequentially. Doing any of these things could cause incorrect statistics reporting, database timeouts in the recover process or the MPS, and/or unnecessary and wasteful rework. It could also result in one recover process trying to clean up data while the other is trying to recover it.

The report file should not be removed until the recover process is complete or detailed information about each file will be lost.

F.96.10. Files

$HPSS_PATH_TMP/recover.PVNAME.TIME.report.txt

Report file for volume PVNAME. Includes detailed reports of the disposition of each file on the damaged volume. The TIME is presented in YYYY.MMDD.HHmmSS format.

F.96.11. See Also

HPSS Admin Guide: Section "HPSS user data recovery: Recovering HPSS files from damaged HPSS volumes".

F.97. rehome_files(7)

F.97.1. Name

rehome_files - Rewrite files with correct storage segment size

F.97.2. Syntax

rehome_files
--volume-name=*pv*
[--global-db-name=*dn*]
[--authenticator-name=*an*]
[--authentication-mechanism=*am*]
[--login-principal=*lp*]
[--db-schema-name=*sn*]
[--subsys-id=*ss*]
[--threads=*nt*]
[--help]

rehome_files
-v *pv*
[-g *dn*]
[-k *an*]
[-m *am*]
[-p *lp*]
[-s *sn*]
[-i *ss*]
[-T *nt*]
[-h]

F.97.3. Description

The rehome_files utility rewrites the data for all files on a disk volume which are not using the correct storage segment size as currently defined by the storage class configuration. It copies the data to new storage segments on the same or a different disk within the same Class of Service and storage class at the same level of the hierarchy. It uses the correct current storage segment size to do the copy.

The disk volume must be at the top of the hierarchy (level 0). The utility rewrites only the data for the level 0 disk. Data at lower levels of the hierarchy is not rewritten. The newly rehomed data at level 0 does not need to be re-migrated to its level 0 migration targets.

Before executing rehome_files, all files on the volume must have already been fully migrated to the level 0 migration targets. The utility will skip any file which has not been fully migrated. If a file is migrated but is subsequently modified, it must be re-migrated before rehome_files will process it.

Files which already are using the correct storage segment size will also be skipped.

The purpose of the utility is to cover a single edge case not supported by the repack utility:

1. New disk media has been introduced for a storage class at level 0 of the hierarchy.
2. The storage segment size in the HPSS storage class configuration is redefined to better suit the new disk media. All new files created in the storage class will now have the new segment size. All pre-existing files in the storage class (those on the old disks) will retain the old storage segment size.
3. Before the old disk media can be retired, its data must be copied to the new media. Repack would normally be the logical choice for this purpose. However, repack will create the new storage segments using the old segment size.
4. The rehome_files utility was therefore introduced to do the copy using the new storage segment size.

The old volumes should be marked read only so that the new data is not written to them.

If a repack has already been attempted on an old volume, then it is possible that the files were repacked to new volumes with the incorrect (old) storage segment size. However, new files created on the new volumes should have the correct (new) storage segment size. The new volume therefore has a mixed set of files: some with the new storage segment size and some with the old. The rehome_files utility can be run against the new volume and will skip any files with the correct storage segment size and rehome those with an incorrect storage segment size.

More than one instance of rehome_files should not be executed concurrently for the same volume. The operation will still succeed, but it will potentially waste a great number of extra processing cycles in the core server for performing sanity checks. These sanity checks are quicker and more efficient when only one copy of the utility is executing per volume. Parallel operation on a single volume is provided instead by the -T option, which specifies how many threads to use while performing the rehome.

If a single file is spread across two or more virtual volumes at level 0, and the rehome_files utility is executed against one of these volumes, it will rehome the data for that file from all of the volumes. It is therefore more efficient under these circumstances to rehome those volumes one at a time, not concurrently. When the first volume is rehomed, the utility will take care of all files on that volume plus any parts of those files which are resident on any of the other volumes. Then if the second and other volumes still have any remaining files (which would be files which did not also have data on the first volume), they can be rehomed separately.



Changes to the max file size allowed by a COS could cause rehome_files to fail.
For example, if a file of size X was written to the COS, and later the COS max file size was changed to something less than X, and we tried to rehome the file, the creation of the new copy of the file would be denied.
In this case, it may be necessary to temporarily increase the COS max file size to allow the rehome.

F.97.4. Options

The rehome_files utility supports the following options:

-g, --global-db-name=*Global DB Name*

Specifies the ASCII name of the global database where the basic configuration information for HPSS is found. If the global database name is not provided in the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used. The default is "cfg."

-h, --help

Specifies that usage help should be displayed.

-i, --subsys-id=*Subsystem ID*

Specifies the ID of the subsystem in which the volume resides. If the global database name is not provided in the command line, the value of the HPSS_SUBSYS_ID environment variable will be used. The default is 1.

-k, --authenticator-name=*Authenticator*

Specifies the authenticator with which this utility will run. If the authenticator name is not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used. The default is "/var/hpss/etc/hpss.keytab." Systems using unix authentication should use "/var/hpss/etc/hpss.unix.keytab."

-m, --authentication-mech=*Mechanism*

Specifies the security mechanism used to authenticate the client. If the security mechanism is not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used. Allowed values are "unix" or "krb5."

-p, --login-principal=*Principal Name*

Specifies the server principal name which this utility will run under. If the principal name is not provided in the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used. The default is "hpssssm."

-s, --db-schema-name=*Schema Name*

Specifies the ASCII name of the Db2 schema that contains the global database (-g). If the schema name is not provided in the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used. The default is "HPSS."

-T, --threads=*nt*

Specifies the numbers of threads used. The default number of threads is 4. The minimum allowed is 1 and the maximum allowed is 1000.

-v, --volume-name=*pv*

Specifies the name of the volume for which to rehome files. The format of the name is the six character volume name followed by two zeros.

F.97.5. Notes

The rehome operation uses standard HPSS internal copy mechanisms, in which no checksum verification is performed. Sites using checksums should first record the existing checksums and then perform hpsssum on all rehomed files to verify that the data still matches the checksums.

F.97.6. Examples

rehome_files -i 1 -v ABCDEF -T 10

Rehome the files for subsystem 1, volume ABCDEF, using 10 threads.

F.97.7. See Also

repack

F.98. remove(7)

F.98.1. Name

remove - HPSS Volume Remove Utility

F.98.2. Syntax

remove --transit-file-name=*fn*
[--subsystem-id=*ss*]
[--authentication-mech=*am*]
[--authenticator-name=*an*]
[--db-schema-name=*sn*]
[--do-not-eject]
[--do-not-export]
[--global-db-name=*dn*]
[--login-principal=*lp*]
[--source-volumes-filename=*fn*]
[--storage-class=*sc*]

remove
-i *ss*
-f *fn*
-m *am*
-k *an*
-s *sn*
-e
-n
-g *dn*
-p *lp*
-S *fn*
-c *sc*

F.98.3. Description

"remove" is a HPSS System Maintenance Utility. It is intended to be used during the process of replacing one storage technology with another. This program will remove a "retired" volume from the Core Server that manages it and optionally export the volume from the PVL. If the volume is exported all metadata records in the PVL and PVR are removed, and the volume optionally may be ejected from a robot.

In order for a volume to be removed, it must be in "retired" state, and the number of storage segments residing on the volume must be zero.

Volume removal candidates can be specified via two methods. The first method allows a list of Core Server physical volumes to be specified in a file. This is the most selective way to remove volumes as specific volumes can be chosen. The second method is to specify the storage class of the volumes to be removed. This is a more general approach and allows a large number of volumes to be retired

during a single run. All volumes in the storage class that are empty and in retired state will be removed.

Removing a volume is done in two steps. In the first, the volume is removed from the Core Server metadata. As part of this process, the Core Server "deallocates" the volume in the PVL that manages it.

The second step is to export the volume from the PVL. This step will be skipped if the --do-not -export command line option is given.

If an error occurs when exporting the volume from the PVL, the name of the volume is written to the transit file. This file can be handed back to remove for additional attempts of the PVL export step. Do this by giving remove the transit file (with the --transit-file-name option) and an empty input file. The program will attempt to export the volumes named in the transit file, but won't remove any other volumes. Each time this is tried, the transit file is re-written with the list of volumes that were not exported. When all have been exported, remove deletes the transit file, leaving nothing to do.

The program is designed to allow an input file to be used in multiple invocations of the program without modification. If some of the volumes listed in the file are not removed in the first attempt, they will be tried in subsequent attempts. Volumes listed in the file that are already removed are ignored. The transit file accumulates a list of volumes that were not exported.

F.98.4. Options

--transit-file-name, -f *fn*

The name of a file which holds volume recovery information. If the PVL export fails the information is saved in this file and an attempt to export the volume will be made during the next run of the remove utility.

--subsystem-id, -i *ss*

The ID of the subsystem that contains the volumes to be removed. HPSS_SUBSYS_ID will be used if an argument is not specified on the command line.

--global-db-name, -g *dn*

The pathname of the global configuration database. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

--db-schema-name, -s *sn*

The name of the database schema that contains the HPSS metadata. If this argument is not specified on the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--login-principal, -p *lp*

The server principal name which this utility will run under. Usually run as SSM. The Principal name of the SSM can be found from SSM on Security Controls tab of the Server Configuration window for SSM. If set, the environment variable HPSS_PRINCIPAL_SSM will be used if an argument is not specified on the command line.

--authenticator-name, -k *an*

The authenticator which this utility will run with. The environment variable HPSS_PRIMARY_AUTHENTICATOR will be used if an argument is not specified on the command line.

--authentication-mechanism, -m *am*

The security mechanism which this utility will use to authenticate the user. The environment variable HPSS_PRIMARY_AUTHN_MECH will be used if an argument is not specified on the command line.

--do-not-export, -n

If specified, the volumes will be removed from the Core Server but will not be exported from the PVL. The volumes will remain in PVL and PVR metadata.

--do-not-eject, -e

If specified, when volumes are exported, they will not be ejected from the library. This option only has an effect when the --do-not-export option is not specified.

--storage-class, -c *sc*

The storage class id of the volumes to be removed. If volumes are to be selected via an input file, do not specify this argument. The Storage Class id can be found from SSM through: Monitor, Active Storage Classes.

--source-volumes-filename, -S *fn*

The name of a file containing a list of Core Server physical volumes to remove. The names must be left justified in the file, one PV name per line. The names must be 8 characters in the style returned by dump_sspvs.

F.98.5. Example

The following command will remove (but not export) the Core Server physical volumes specified in the file "remove_list". The volumes listed in the "remove_list" must be managed in the storage subsystem specified by the --subsystem-id argument.

```
    $ remove --subsystem-id=1 --transit=transit_file --do-not-export
--source=remove_list
```

The contents of the file remove_list contains the SS PV names for four volumes. Example:

```
      $ cat remove_list
      BCK05900
      BUD13200
      BCK06000
      BCK06100
```

The following command will remove (and export) volumes in retired state which are managed by

the Core Server in subsystem 2 with storage class 5. The names of any PVs not exported are stored in "transit_file".

```
    $ remove --subsystem-id 2 --transit transit_file --storage-class 5
```

F.98.6. See Also

retire, repack

F.99. repack(7)

F.99.1. Name

repack - HPSS Volume Repack Utility

F.99.2. Syntax

Command line form 1 - repack selects tape volumes to repack

repack --storage-class=*cl*
--number-to-repack=*n*
[--subsystem-id=*ss*]
[--authentication-mechanism=*am*]
[--authenticator-name=*an*]
[--db-schema-name=*sn*]
[--destination-tapes-filename=*fn*]
[--global-db-name=*dn*]
[--include-checked-out-volumes]
[--login-principal=*lp*]
[--max-bytes-per-aggregate=*ba*]
[--break-ordered-aggregates]
[--no-aggregate-single-files]
[--no-dest-tape-segregation]
[--reduce-tape-positioning]
[--eliminate-whitespace]
[--reduce-tape-pos-detection]
[--reduce-tape-pos-aggr=*ap*]
[--reduce-tape-pos-sparse=*sp*]
[--select-degraded-rait]
[--select-file-family=*fm*]
[--select-retired-tapes]
[--selection-threshold=*st*]
[--verbose]
[--notification-interval=*ni*]

repack -c *cl*
-n *n*
[-i *ss*]
[-m *am*]
[-k *an*]
[-s *sn*]
[-D *fn*]
[-g *dn*]
[-o]
[-p *lp*]
[-L *ba*]
[-b]

[-F]
[-T]
[-P]
[-W]
[-0]
[-1 *ap*]
[-2 *sp*]
[-d]
[-f *fm*]
[-M]
[-v *st*]
[-V]
[-I *ni*]

Command line form 2 - user supplies a list of disk or tape volumes to repack

repack --volume-type=*vt*
--source-volumes-filename=*fn*
[--subsystem-id=*ss*]
[--authentication-mechanism=*am*]
[--authenticator-name=*an*]
[--db-schema-name=*sn*]
[--destination-tapes-filename=*fn*]
[--global-db-name=*dn*]
[--include-checked-out-volumes]
[--login-principal=*lp*]
[--max-bytes-per-aggregate=*ba*]
[--break-ordered-aggregates]
[--no-aggregate-single-files]
[--no-dest-tape-segregation]
[--reduce-tape-positioning]
[--eliminate-whitespace]
[--reduce-tape-pos-detection]
[--reduce-tape-pos-aggr=*ap*]
[--reduce-tape-pos-sparse=*sp*]
[--verbose]
[--notification-interval=*ni*]

repack -t *vt*
-S *fn*
[-i *ss*]
[-m *am*]
[-k *an*]
[-s *sn*]
[-D *fn*]
[-g *dn*]
[-o]
[-p *lp*]
[-L *ba*]

[-b]
[-F]
[-T]
[-P]
[-W]
[-0]
[-1 *ap*]
[-2 *sp*]
[-V]
[-I *ni*]

F.99.3. Description

"repack" is an HPSS System Maintenance Utility that consolidates data on disks and tapes. When tapes are to be repacked, the program can be invoked through an SSM screen, or at a command line. To repack disks, or to repack specifically named tape volumes, the program must be invoked at a command line.

Repack's function is to cause the Core Server to move storage segments from sparse volumes to other volumes in the same storage class and family. When tapes are emptied with repack, they can be reclaimed and returned to service, increasing the number of free tapes in the system. Repack can also be used to migrate disk and tape storage segments from old media to new media.

Two distinct forms of the command line exist. In the first, repack selects volumes to process based on storage class and other selection criteria. This command line form can only be used to repack tapes. Disks cannot be repacked in this fashion.

In the second form, repack reads a text file that contains the names of Physical Volumes to be repacked. If the volumes are striped, only one PV name from each virtual volume should be put in the input file. The name of any PV that is part of the VV is sufficient. PV names in the file are 6 characters long, left adjusted, one per line.

When command line form 2 is used, the volumes listed in the input file can be of mixed storage classes and families. However, they must all be disks or all tapes and must match the type given in the --volume-type argument.

When the --destination-tapes-filename option is provided, the program opens the file named by the option and reads target tape names from it. The tape names must be provided one per line, left adjusted. They must be 6 characters long in the style that is standard for HPSS. This file will be ignored when repacking disks.

When command line form 1 is used, repack will select tapes in RO and EOM Condition. Tapes in RW Condition which are also Retired will be treated as if they were in RO Condition. Tapes in RWC or DOWN condition cannot be repacked. Tapes in EMPTY Condition cannot be repacked using this command line form.

When command line form 2 is used to repack tapes, the named tapes may be in RW, RO, EOM or EMPTY Condition.

When repacking disks, the selected volumes must be in RW, RO, or EMPTY Condition. Disks in RWC

or DOWN Condition cannot be repacked.

Repack will never process a disk or tape volume in RWC or DOWN Condition.

When repacking tapes, the program removes unused portions of file aggregates it finds in the source tapes. This is done by copying the active parts of the aggregates and leaving the inactive parts behind. The program also creates new aggregates on the output tape by bringing together smaller tape storage segments from the input tape and writing new aggregates to the output tape. The --max-bytes-per-aggregate command line argument controls how this is done.

Values for the global database name, schema name, authentication mechanism, authenticator and principal name will be taken from the user's environment if the corresponding arguments are not supplied on the command line. If these environment variables do not appear in the user's environment, default values will be used. The command line arguments override the environment. The user must supply the --subsystem-id command line argument. There is no default value.

F.99.4. Options

--authentication-mechanism, -m *am*

The security mechanism used to authenticate the client. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--authenticator-name, -k *an*

The authenticator which is used to obtain credentials. If not provided on the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--db-schema-name, -s *sn*

The ASCII name of the Db2 schema that contains the global database. If not specified on the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--destination-tapes-filename, -D *fn*

The pathname of a file which contains the Core Server physical volume names of tapes to be written by repack. These tapes will be considered in the order they appear in the file. Tapes that cannot be written for any of several reasons will be skipped. No error message is generated when this occurs. If the supply of destination tapes runs out before repacking is finished, repack stops with an error.

--global-db-name, -g *dn*

The ASCII name of the global database where the basic configuration information for HPSS is found. If not specified on the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--include-checked-out-volumes, -o

Tape volumes that are checked-out (shelved) will be included in the repack. By default, checked-out tapes are not included. This option can be used in both command line forms.

--login-principal, -p *lp*

The server principal name which this utility will run under. Repack is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server

Configuration window for SSM. If not provided on the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--max-bytes-per-aggregate, -L *ba*

Controls the maximum number of bytes of data from the source tape that will be placed in each output aggregate. This value is an integer and optional unit (example: 10G is 10*1024*1024*1024). Units are K, M, G, T, P, E and case insensitive. The units can have an optional B appended (as in KB), though this does not affect behavior. If this value is set to 0, output aggregation is turned off and input tape storage segments are moved to the output tape one at a time. The maximum value is the VV Estimated Size for the corresponding storage class. This value is ignored when repacking disks.

--break-ordered-aggregates, -b

Controls whether aggregates which are ordered by some criteria (other than creation time) should be eligible to be broken up rather than repacked together. Enabling this flag will allow aggregates which were written and ordered (e.g. by parent directory) to be split up or joined with other data on the tape which does not share the same ordering criteria. By default, repack will move aggregates which are ordered intact with no changes aside from removal of empty space.

--no-aggregate-single-files, -F

By default, single files are included in aggregates on the Repack output tape. This is usually accomplished by a.) combining multiple sequential single files on the source tape into a new aggregate or b.) accumulating one or more single files into a nearby aggregate on the source tape so that they're all in one aggregate on the output tape. Use this option to disable the combination of single files into aggregates. With this option, single files on the source tape will remain single files on the output tape.

--no-dest-tape-segregation, -T

For this instance of Repack, ignore the value of the HPSS_CORE_REPACK_OUTPUT environment variable. Even if the variable is set to segregate Repack data, don't segregate the data for this instance of Repack. Instead, for this instance of Repack, any tape may be chosen as a destination (or output) tape. See the ENVIRONMENT section below for more details about the HPSS_CORE_REPACK_OUTPUT environment variable.

--reduce-tape-positioning, -P

Controls whether aggregates on the source volumes are reorganized by repack or moved in full. By default, repack will break apart aggregates on the source, squeezing out any inactive space, and may also append additional data from the source onto the aggregate. This activity allows Repack to be efficient with destination tape space, however this process can become expensive in situations where the tape is sparse, leading to poor overall repack throughput. This option disables that processing, causing all segments to be moved as-is, for improved repack processing speeds at the cost of destination storage space. This argument is not valid when repacking disks.

--eliminate-whitespace, -W

This is the default tape repack behavior. This flag will ensure that no inactive space within a tape aggregate will be copied to the Repack output tape. This also allows for the creation of new aggregates.

--reduce-tape-pos-detection, -0

This flag enables the --reduce-tape-positioning method of repacking when the source tape meets certain criteria. These criteria are the percentage of the active data on tape that is within a tape aggregate, and the percentage of the active data compared to the total written length of the tape (a rough measure of sparsity).

By default, these two parameters are 25% and 50%, respectively. So the active data must be at least 25% aggregate data, and the tape must be at least 50% sparse for this method to flip on. These parameters can be fine-tuned using the --reduce-tape-pos-aggr and --reduce-tape-pos -sparse options.

--reduce-tape-pos-aggr, -1 *ap*

Controls the percentage of total active data that must reside in aggregates in order for the reduced tape position heuristic to enable that method of repack.

--reduce-tape-pos-sparse, -2 *sp*

Controls the percentage of inactive bytes that must exist on a tape in order for the reduced tape position heuristic to enable that method of repack.

--number-to-repack, -n *n*

When repack selects tape volumes to repack (command line form 1), this is the number of tape volumes to select. The rest of the selection options are used to further refine the selection. This argument is invalid in command line form 2.

--select-degraded-rait, -d

When repacking a RAIT tape storage class, or RAIT tape volumes named in an input file, select only the volumes that the system has marked as being degraded in some way. Degraded volumes will have one or more physical volumes that cannot be mounted, one or more PVs that cannot be read, or one or more PVs that cannot be written.

--select-file-family, -f *fm*

When repack selects tape volumes to repack (command line form 1), it will select tape volumes only in the given file family. This option is invalid in command line form 2.

--select-retired-tapes, -M

When repack selects tape volumes to repack (command line form 1), only tape volumes in "retired" state will be selected. The volumes must be in RW, RO, EMPTY or EOM condition. This option is invalid in command line form 2.

--selection-threshold, -v *st*

When repack selects tape volumes to repack (command line form 1), only tapes whose occupancy level is equal to or less than this value will be selected. "st" is interpreted as an integer percentage. The value 35 is interpreted as 35%. The occupancy of candidate tape volumes is calculated by dividing the number of data bytes remaining on the volume by the original number of bytes written to the volume. Set this value to 100 to disable selection by occupancy level. The default value for this argument, which applies only to command line form 1, is 40%. This option is invalid in command line form 2.

--source-volumes-filename, -S *fn*

The pathname of a file which contains the names of the physical volumes to repack. The names of either disk or tape volumes may be given, but they may not be mixed and must match the type given in --volume-type. The name of any of the physical volumes in a striped disk or tape volume is sufficient to select the volume. This option is invalid in command line form 1.

--storage-class, -c *cl*

When repack selects tape volumes to repack (command line form 1), only tapes in this storage class will be selected. This argument is required in form 1 of the command line, and is invalid in form 2.

--subsystem-id, -i *ss*

The id of the subsystem in which the volumes reside. This is always non-zero positive number. If not provided on the command line, the value of the environment variable HPSS_SUBSYS_ID will be used.

--verbose, -V

Selects the verbose output mode. Progress messages are sent to standard output on about 5 minute intervals indicating the number of segments that have been repacked and the number of errors that have occurred. If a segment takes more than 5 minutes to repack, the progress message will appear when the segment is finished.

--notification-interval, -I *ni*

How often to log status information, in seconds. This prevents notifications from being logged too often. The default is 300 seconds.

--volume-type, -t *vt*

When repack reads a list of disk or tape volumes to repack from an input file (command line form 2), this is the type of those volumes. The value must be "disk" or "tape". Volumes of different types may not be mixed in the input file.

F.99.5. Selection Criteria

When repack selects tapes to repack (command line form 1), the selection criteria are logically "anded" together. The criteria are:

1. Storage Class
2. File Family
3. Retired State
4. Degraded RAIT State
5. Check-in State
6. Occupancy

The first selection criterion is Storage Class. All volumes in the given Storage Class are selected. Tapes that are in any sort of writable state, EMPTY or DOWN are removed from the list.

Then, if the --select-file-family option was provided, any volumes not in the given file family are

removed from the list.

Then, if the --select-retired-tapes option was provided, any volumes not in "retired" state are removed from the list.

Then, if the --select-degraded-rait option was provided, any volumes that are not degraded RAIT volumes are removed from the list.

Then, if the --include-checked-out-volumes option was not selected, any volumes that are checked out are removed from the list.

Finally, any volumes who occupancy level is greater than the value given in --selection-threshold, or greater than 40% if the option was not provided, are removed from the list.

F.99.6. Environment

The HPSS_CORE_REPACK_OUTPUT environment variable enables or disables the tape repack output segregation feature. If this environment variable is not present in the system's environment, or is present and has the value "on", files moved from one tape VV to another will be written to tapes that contain only files written by repack. If this variable has the value "off", files moved from one tape VV to another may be mixed on with files that were written by the migration subsystem, or directly to tape by users. This environment variable can be turned off for a single Repack instance with the --no-dest-tape-segregation command line option. See above for details.

F.99.7. Notes

Physical volume names in the optional input and output files can be either 6 or 8 characters long. Strictly speaking, HPSS uses 8 character volume names, but has always required that the last two characters be zeros. Repack will accept the more conventional 6 character names and will append two zeros before using them.

The program accepts both long and short command line argument names on LINUX systems, but only short names on AIX systems. This is due to a limitation of the AIX programming environment. Long names can be abbreviated to the smallest number of characters that makes the name unique.

When repack selects tapes to be repacked, EMPTY tape will not be chosen. However, if the user uses command line form 2 and names an EMPTY tape in the input file, repack will process it. It is unlikely any data will be found on the tape, but this feature is provided as a means of cross-checking the system.

F.99.8. Example

The following command will repack 5 tape volumes specified in the file "vv_repack_file". The Core Server that services the volumes is in subsystem 1.

Contents of the file vv_repack_file:

```
cat vv_repack_file
BUD132
```

```
     BUD133
     BUD134
     BUD135
     BUD136

     $ repack --subsystem-id=1 --volume-type=tape --source-volumes
-filename=vv_repack_file
```

The following command repacks one disk volume specified in the file "disk_repack.src". The volume is managed by subsystem 1.

Contents of the file disk_repack.src:

```
     cat disk_repack.src
     SSA010

     $ repack -i 1 -t disk -S "./disk_repack.src"
```

The following command will repack 10 "retired" tape volumes with less than 20% data in use in storage class 3.

```
     $ repack --subsystem-id=1 --number-to-repack=10 --storage-class=3 \
              --selection-threshold=20 --select-retired-tapes
```

The following command will repack 5 tape volumes in storage class 3 and family 7 to volumes of the same class and family.

```
     $ repack --subsystem-id=1 -number-to-repack=5 --storage-class=3 \
              --select-file-family=7
```

The following command repacks 10 retired tape volumes in storage class 7. The selection threshold is not considered, tapes with any amount of active data will be selected. Shelved tapes will also be included.

```
     $ repack -i 1 -n 10 -c 7 -v 100 -M -o
```

The following command repacks up to 25 Rait volumes that have been marked as "degraded" in some way, from storage class 8 in subsystem 1. The selection threshold is set to 100, so that all volumes meeting these selection criteria will be repacked.

```
     $ repack --storage-class=8 --subsystem-id=1 --number-to-repack=25 \
              --select-degraded-rait --selection-threshold=100
```

F.99.9. See Also

reclaim, retire, remove

F.100. retire(7)

F.100.1. Name

retire - HPSS Volume Retire Utility

F.100.2. Syntax

retire [--subsystem-id=*ss*]
[--global-db-name=*db*]
[--db-schema-name=*sn*]
[--login-principal=*lp*]
[--authenticator-name=*an*]
[--authentication-mech=*am*]
[--storage-class=*sc*]
[--select-vv-condition=*vc*]
[--unretire]
[--source-volumes-filename=*fn*]

retire [-i *ss*]
[-g *HPSS Global Database Name*]
[-s *Database Schema Name*]
[-p *Utility Principal Name*]
[-k *Utility Authenticator Name*]
[-m *Utility Security Mechanism*]
[-c *Storage Class ID*]
[-C *VV Condition*]
[-u]
[-S *Input File Name*]

F.100.3. Description

retire is an HPSS System Maintenance Utility. Its intended use is to facilitate replacing one storage technology with another by setting the "retired" flag in the storage maps of obsolete technology volumes. New storage segments will not be created on retired volumes. Retired volumes may be removed from the system by repacking all of the useful information on them to other volumes, leaving the retired volumes empty. These volumes can then be removed from the system with the "remove" utility.

Volume candidates for retirement can be specified via two methods. The first method allows a list of Core Server physical volume names to be specified through standard input. This is the most selective way to retire volumes as specific volumes can be chosen.

The second method is to specify both the storage class and volume condition of the volumes to be retired. This is a more general approach and will allow for a large number of volumes to be retired during a single run.

F.100.4. Options

--global-db-name, -g *Global DB Name*

The ASCII name of the global database where the basic configuration information for HPSS is found. If not specified in the command line, the value of the HPSS_GLOBAL_DB_NAME environment variable will be used.

--db-schema-name-sn, -s *Schema Name*

The ASCII name of the Db2 schema that contains the global database (-g). If not specified in the command line, the value of the HPSS_MM_SCHEMA_NAME environment variable will be used.

--login-principal, -p *Principal Name*

The server principal name which this utility will run under. Retire is usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided in the command line, the value of the HPSS_PRINCIPAL_SSM environment variable will be used.

--authenticator-name, -k *Authenticator*

The authenticator with which this utility will run. The Keytab name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

--authentication-mech, -m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

--subsystem-id, -i *Subsystem ID*

The id of the subsystem in which the volumes reside. If not provided in the command line, the value of the HPSS_SUBSYS_ID environment variable will be used.

--storage-class, -c *Storage Class ID*

The storage class id of the volumes to be retired. The Storage Class id can be found from SSM through: Monitor, Active Storage Classes. This argument cannot be used with -S.

--select-vv-condition, -C *VV Condition*

Restricts the selection of volumes to retire by storage class and VV condition. This argument is a text string and is one of the following:

State	Description
"rwc"	Select volumes in read-write-create condition.
"RWC"	Select volumes in read-write-create condition.
"rw"	Select volumes in read-write condition.
"RW"	Select volumes in read-write condition.
"ro"	Select volumes in read-only condition.

"RO"	Select volumes in read-only condition.
"eom"	Select volumes in EOM condition.
"EOM"	Select volumes in EOM condition.
"empty"	Select volumes in empty condition.
"EMPTY"	Select volumes in empty condition.
"down"	Select volumes in down condition.
"DOWN"	Select volumes in down condition.

This argument can be specified only when -c is specified and cannot be used with -S.

--source-volumes-filename, -S *File*

Is a file containing a list of the Core Server PV names of volumes to be retired, one to a line. The PV names must be in HPSS internal format, i.e. they must be 8 character names — the first six are the conventional PV name — the last two characters must be "00".

F.100.5. Example

The following command will retire the volumes specified in the file "retire_list" found in subsystem 1.

```
$ retire --subsystem-id=1 --source-volumes-filename=retire_list
```

The contents of the file retire_list contains the Storage Server physical volume names for 4 virtual volumes. four volumes. Example:

```
$ cat retire_list
BCK56900
BUD13000
BUD13500
BCK65000
```

The following command will retire volumes in subsystem 1 and in storage class 14:

```
$ retire -i1 -c14
```

The following command will retire EOM volumes in subsystem 2 and storage class 14:

```
$ retire --subsystem-id=1 -storage-class=14 --select-vv-condition=EOM
```

F.100.6. See Also

remove, repack

F.101. rq_cleanup(7)

F.101.1. Name

rq_cleanup - Clean up persisted read queues

F.101.2. Synopsis

rq_cleanup [--help] [--identifier *IDENTIFIER*]
[--all] [--subsys-db-name *subsysname*] [--schema *SCHEMA*]
{read, stage}

F.101.3. Description

This tool provides a method to remove persisted read queues and stage callbacks (either all or a specific read queue or stage callback) from metadata. This should only be run when the core server is down in cases where the queued read or stage callback operations should not be loaded back upon core server restart. Otherwise, this will result in the loss of client reads and stages.

The tool processes either a specific read queue, stage callback, or all read queues, or all stage callbacks on a single subsystem. For systems with multiple subsystems, the tool needs to be run against each subsystem.

F.101.4. Notes

Note that this is a metadata-only operation. No HPSS data transfers occur as part of this tool's operation.

The HPSS system should be stopped. Otherwise, spurious HPSS metadata errors might appear attempting to remove metadata about persisted read queues and stage callbacks.

F.101.5. Required Arguments

queue_type

Specifies the queue type (*read* or *stage*) for the cleanup operation

F.101.6. Optional Arguments

-h, --help

show a help message and exit

--all

All read queue contexts or all stage callbacks

--identifier IDENTIFIER

Read queue context or stage callback ID. The value may be specified in two different formats. One is the format as stored by Db2, for example, a3141e31654379459364f9701a912e65. The other format is used by tools such as lori and the core SIGHUP output, for example, 311e14a3-4365-

4579-9364-f9701a912e65.

--subsys-db-name subsys1

The name of the subsystem to process (default: *subsys1*)

--schema SCHEMA

database schema name (default: *hpss*)

F.101.7. Prerequisites

The following software prerequisites are required for the rq_cleanup tool:

- Python 3.6 or later
- the ibm-db Python package (available via *pip install ibm_db* or here: https://pypi.org/project/ibm-db/)

F.102. rtmu(7)

F.102.1. Name

rtmu - retrieve and format Realtime Monitoring information

F.102.2. Syntax

rtmu [-d *Server Descriptive Name*]
[-s *Server Type*]
[-i *Request Id*]
[-f *File Name*]
[-v *Physical Volume Name*]
[-a *Action*]
[-o *Output Format*]
[-t *Time Interval*]
[-n *Number of Calls*]
[-p *Principal Name*]
[-k *Authenticator*]
[-m *Authentication Mechanism*]
[-g *Database Name*]
[-h *Schema Name*]
[-r]
[-j]
[-z *Debug Flag*]
[-?]

F.102.3. Description

The rtmu utility is used to retrieve and format real-time monitoring information from selected servers. Currently, the only servers that maintain information about requests are the Core, Mover, Rait and Gatekeeper servers. Access to this request information can aid the administrator or support personnel in determining the state of particular requests. The entries returned by the servers are grouped by request ID and printed to stdout in the selected format. Each RTM entry returned contains information such as the following: request id, type of request, state of request, server descriptive name, server id, start time of request, time request information was last updated, id of the thread handling the request, process id handling the request, the reason the request is waiting, the resource being waited on (e.g. server, device, volume), and server specific information.

The server specific information returned (depending on entry type) is as follows:

Core

subsystem id, realm id, fileset id, fileset name, bitfile id, bitfile pathname, user id, group id, COS id, Storage Class id, Bitfile Service bytes requested, Bitfile Service bytes moved, PVL job id, Mover id, device id, segment, Virtual Volume, physical volume name, Storage Service bytes requested, Storage Service bytes moved, current relative position, current absolute position, excess VVs in use, Storage Class id, and family id

Mvr

initial transfer offset, current transfer offset, device id, volume id, bytes requested, and bytes moved

Rait

initial transfer offset, current transfer offset, bytes requested, bytes moved, and parity index.

GK

authorized caller, bitfile id, connection id, client connection id, control number, realm id, group id, host address, open flag, request type, stage flags, stage length, stage offset, stage storage level, user id, and wait time

Not all server specific information is defined in each RTM record and only values that have been defined by the specific server are reported.

There are three output format options: short, medium and long. In all formats, a user is identified by login name (if possible) rather than uid, and files are referenced by their HPSS pathnames rather than their bitfile IDs. The data is grouped by request ID and printed in order by the age of the request: oldest first. All of the formats provide the following minimal amount of information: the server holding the oldest RTM record with this request ID and its age, along with the server that most recently updated an RTM record with this ID and how long ago that happened.

The "short" output is in a fixed-width tabular form with one line per unique request ID returned by the servers. This is the default output format and can be used to view a snapshot of the waiting requests. The "medium" format contains more (but not all) data returned by the servers. It consists of a series of stanzas, one for each request ID returned by the servers. Each stanza contains several lines of output from each server returning data with this request ID. Only the more important values (in abbreviated format) are printed. The "long" format is basically a formatted structure dump of all defined RTM data returned by all of the queried servers. This option can produce a large amount of output and is structured in such a way that it can be easily parsed by a program or PERL script. Field definitions for all three formats are given at the end of this man page.

A given server may return multiple entries from a single request. For example, the Core Server could have eight entries to describe 8 waits for mounts on a write to a 8-way tape Class of Service.

Retrieval options are supported for a specified request id, HPSS file name, or physical volume name. These options are mutually exclusive. If -f (File Name) is entered, a Core Server Descriptive Name must be supplied. If -v (PV Name) is entered, a Core Server Descriptive Name must also be entered.

If a command is supplied without the -n option, then a single real-time request is made. Otherwise, the number of specified calls will be made at the interval specified by the -t option. If -t is not specified, an interval of 30 seconds will apply.

The user should specify which servers to query for information. Servers can be specified by descriptive name (using the -d option), and/or by server type (using the -s option). All servers can be specified by using the "-s all" option. By default, if no servers are specified, all servers will be queried. If a center runs a large number of servers (usually Movers), querying all of them can take a while. In this case it may be beneficial to use the recursive (-r option). With this option, only the

core servers are queried (initially). If any resulting RTM requests are waiting on other servers, those servers will also be queried. That is, only the servers required to complete the information on each request are queried.

The rtmu utility must run as an authorized principal. Typically, the utility would be expected to run on behalf of the SSM principal and have access to the server keytab file. In addition, the utility requires access to the metadata database to access information. It is recommended that the proper HPSS environment variables be configured before using this utility. Otherwise, principal name, keytab file name, database name, and schema name must be supplied as inputs or defined as environment variables.

F.102.4. Options

-d *Descriptive Name*

The descriptive name of the server. If specified, RTM information will be retrieved for the requested server. If not specified, the request will apply to all servers.

-s *Server Type*

The type of server (core, mover, rait, gatekeeper, all). If specified, RTM information will be retrieved from all servers marked as executable in the configuration database of the specified type. By default, if no servers are specified, all servers will be queried.

-i *Request Id*

The request id for the request. If specified, RTM information will be for entries matching the request id.

-f *File Name*

The name of an HPSS file. If specified, RTM information will be for Core Server entries where the bitfile id corresponds to the file name. A Server Type of "core" or the Descriptive Name for a Core Server must be specified.

-v *Physical Volume*

The name of a physical volume. If specified, RTM information will be for Core Server entries associated with the volume name. A Server Type of "core" or the Descriptive Name for a Core Server must be specified.

-a *Action*

This is name of an action. If specified, RTM information will be filtered by entries that match the specified action.

-o *Output Format*

Specified the type and verbosity of the RTMU output. The options are: "short", "medium" and "long". The default output format is "short".

-t *Time Interval*

The number of seconds between RTM retrievals. The -n option specifies the number of retrieval passes. If not specified, the default is 30 seconds.

-n *Number of Calls*

The number of times RTM information is retrieved. If specified, RTM information will be retrieved every -t seconds for the number of calls indicated. If not specified, the default is 1 retrieval.

-p *Principal Name*

The server principal name which this utility will run under. Usually run as SSM. The Principal name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

-k *Authenticator*

The authenticator which this utility will run with. Usually run as SSM. The Keytab name of the SSM can be found from SSM on the Security Controls tab of the Server Configuration window for SSM. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

-m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHENTICATOR will be used.

-g *Database Name*

The name of the metadata database. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

-h *Schema Name*

The name of the database schema. If set, the environment variable HPSS_MM_SCHEMA_NAME will be used if an argument is not specified on the command line.

-r

Option to recursively call servers. With this option, any server listed as the resource being waited on will also be called for RTM retrieval. If not specified, the default is no recursion. A core server (using the -d or -s options) must be specified.

-j

Output in JSON format.

-z *Debug Flag*

Debug flag that can be used to provide additional information. A value of "0x00040000" will print a list of servers being used.

-?

Option to output usage (help) information.

F.102.5. Examples

To retrieve entries from all configured servers in short (tabular) format:

```
$ rtmu
or
$ rtmu -s all
```

To retrieve entries for volume A00100 in medium format:

```
$ rtmu -d "Core Server" -v A00100 -o medium
or
$ rtmu -s core -v A00100 -o medium
```

To retrieve entries for file /home/bob/myfile:

```
$ rtmu -d "Core Server" -f "/home/bob/myfile"
or
$ rtmu -s core -f "/home/bob/myfile"
```

To retrieve entries for request id 10500 from all servers in medium format:

```
$ rtmu -s all -i 10500 -o medium
```

To retrieve entries for all core servers each second for 10 intervals:

```
$ rtmu -n 10 -t 1 -s core
```

To retrieve entries from Core Server and recursively from any servers which are resources being waited on:

```
$ rtmu -d "Core Server" -r
or
$ rtmu -s core -r
```

To retrieve entries for action "write":

```
$ rtmu -a "write"
```

F.102.6. Short Format Field Definitions

When selecting the short output format ("-o short" option, the default) the RTM data will be presented in tabular form with one row per unique Request ID. The entries will be sorted by the age of the request, oldest to youngest. This format gives a brief snapshot of the state of the waiting requests in the system. The definitions of the fields (columns) are:

Column Name	Description
RequestId	The ID of the request.
Oldest Start	The age, or time difference between when the oldest request with this ID entered the system and when the report was generated. Both the name of the server and the age is displayed.
Newest Update	The age, or time difference between when a server most recently updated an RTM record with this ID and when the report was generated. Both the name of the server and the age is displayed.
State	The state of the requested operation. Examples include "in progress", "suspended" and "blocked".
Action	The action or operation this request is currently executing. Examples include "Mover write", "PVL verify" and "write tm" (write tape mark).
User	The username associated with this request (if available).
File	The filename associated with this request (if available). Due to size restrictions, only the trailing portion of the pathname may be displayed.
PVName	The name of the Physical Volume associated with this request (if available).
PVLJobId	The PVL Job Id (if available).
Wait Reason	A (truncated) text string indicating what the request is waiting for.

F.102.7. Medium Format Abbreviations

When selecting the medium output format ("-o medium" option) the stanzas for each request will contain abbreviations to reduce the volume of output. The definitions for each abbreviation are given below:

Core Server Requests

Name	Description
sysid	Subsystem ID of this Core servers.
cos	Class of Service associated with this request.
sc	Storage Class associated with this request.
bytes_req	Number of Bytes requested.

bytes_moved	Number of Bytes that have been moved.
pvljob	PVL Job ID.
pvname	Name of the Physical Volume associated with this request.
relpos	Physical volume current Relative Address.
abspos	Physical volume current Absolute Address.
excessVV	Waiting for this number of tape Virtual Volumes to become available.
fam	File family.
ss_sc	Storage Class ID of the storage segment.
ss_req	Bytes requested according to the Storage Server.
ss_moved	Bytes moved according to the Storage Server.

Mover Server Requests

Name	Description
init_offset	Initial offset of request.
cur_offset	Current offset of request.
dev_id	Device ID.
vol_id	Volume ID.
bytes_req	Number of Bytes the Mover was requested to move.
bytes_moved	Number of Bytes the Mover has moved.

RAIT Server Requests

Name	Description
init_offset	Initial offset of request.
cur_offset	Current offset of request.
bytes_req	Number of Bytes requested to move.
bytes_moved	Number of Bytes moved.
parity_buf	The parity computation buffer index.

Gatekeeper Server Requests

Name	Description
auth_caller	Authorized Caller
uid	User ID associated with the request.
realm	Realm ID associated with the request.

gid	Group ID associated with the request.
req	Request type ("Create", "Open" or "Stage").
oflag	Open flags of an "Open" request.
stage_lev	Storage Class Level associated with a "Stage" request.
stage_flags	Flags associated with a "Stage" request.
stage_len	Bytes requested in a "Stage" request.
stage_off	Offset of a "Stage" request.

F.102.8. Long Format Field Definitions

When selecting the long output format ("-o long" option) every field (with a defined value) of each RTM record is printed. The output resembles a C-style structure dump. As with the medium output format, the requests are grouped by Request ID and ordered from oldest to youngest. This format can produce a lot of output. You will generally use this when you want the most complete data available on a particular request. That is, you will generally use this format with the "-i", "-v" or "-f" selector options. The field definitions are given below. The Summary section pulls together data from all the RTM records with the given Request ID. The Common sections defines fields that are common to the Core, Mover and GateKeeper RTM records. The server specific Core, Mover and GateKeeper fields follow. In these sections, only fields that contain valid data are printed.

Table 41. Summary Fields

Name	**Description**
ReqId	The Id of this request
ReqCode	The action or operation this request is currently executing. Examples include "Mover write", "PVL verify" and "write tm" (write tape mark).
ReqState	The state of the requested operation. Examples include "in progress", "suspended" and "blocked".
OldestStartDelta	The age, or time difference between when the oldest request with this ID entered the system and when the report was generated.
NewestUpdateDelta	The age, or time difference between the most recent time a server updated an RTM record with this ID and the time when the report was generated.
OldestServer	Name of the server responsible for the OldestStartDelta.
NewestServer	Name of the server responsible for the NewestUpdateDelta.

UserName	The name of the user associated with this request.
PathName	The pathname of the hpss file associated with this request. The reporting level of the path name is controlled by the environment setting HPSS_RTM_USE_FULL_PATH. Refer to the entry for HPSS_RTM_USE_FULL_PATH in the *HPSS Installation Guide* Appendix hpss_env_defs.h for details.

Table 42. Common Fields

Name	Description
ServerDescName	The descriptive name of the server that holds this RTM request record.
StartTimeDelta	The age of this request since it entered the server. That is, the time difference between when the request entered the server and when this report was generated.
UpdateTimeDelta	The time difference between when this server last updated this RTM record and when this report was generated.
ThreadId	The ID of the thread processing this request.
ProcessId	The ID of the UNIX process that is processing this request.
WaitString	A text string indicating what the request is currently waiting for.

Table 43. Core Server Specific Fields

Name	Description
SubsystemId	The ID of the HPSS subsystem owning this request.
UserRealmId	The Realm ID of the user associated with this request.
FilesetId	The ID of the fileset of the file associated with this request.
FilesetName	The text string name of the fileset of the file associated with this request.
BfId	The Bitfile ID of the file associated with this request.
BfPathname	The HPSS pathname of the file associated with this request.

UserId	The ID of the user associated with this request.
UserGId	The Group ID of the user associated with this request.
COSId	The Class Of Service ID of the file associated with this request.
BFS_SClassId	The Storage Class ID known to the bitfile server of the file segment associated with this request.
BFS_BytesRequested	The number of bytes to be moved in this request as known by the bitfile server.
BFS_BytesMoved	The number of bytes already moved for this request as known by the bitfile server.
PVLJobId	The ID of the PVL job associated with this request.
MvrId	The ID of the Mover this request is currently waiting on.
DeviceId	The Device this request is currently waiting on to complete a data move operation.
Segment	The ID of the storage segment being operator on by this request.
VV	The ID of the virtual volume associated with this request.
PVName	The name of the Physical Volume associated with this request.
CurrentRelPosition	The current media position of the associated physical volume, presented as a relative address.
CurrentAbsPosition	The current media position of the associated physical volume, presented as an absolute address.
ExcessVVsInUse	The Tape virtual volumes in use by this storage class have exceeded its quota. Waiting for this number of virtual volumes to become available.
FamilyId	The ID of the File Family of the file associated with this request.
SS_SClassId	The Storage Class Id known to the Storage Server of the file segment associated with this request.
SS_BytesRequested	The number of bytes to be moved by this request as known by the storage server.
SS_BytesMoved	The number of bytes that have already been moved for this request as known by the storage server.

Table 44. Mover Server Specific Fields

Name	Description
InitialTransferOffset	The offset of the start of the mover device for the transfer associated with this request.
CurrentTransferOffset	The current offset of the mover device for the transfer associated with this request.
DeviceId	The ID of the device associated with this request.
VolumeId	The name of the volume associated with this request.
BytesRequested	The number of bytes the mover has been asked to transfer for this request.
BytesMoved	The number of bytes the mover has already transferred for this request.

Table 45. Rait Server Specific Fields

Name	Description
InitialTransferOffset	The offset of the start of the transfer associated with this request.
CurrentTransferOffset	The current offset of the transfer associated with this request.
BytesRequested	The number of bytes the rait engine has been asked to transfer for this request.
BytesMoved	The number of bytes the rait engine has already transferred for this request.
ParityBufferIndex	The index number of the buffer that the parity thread is currently computing.

Table 46. Gatekeeper Specific Fields

Name	Description
AuthorizedCaller	Indicates whether the caller is authorized or not.
BitFileId	The Bitfile ID of the file associated with this request.
ConnectionId	The ID of the (core server) connection associated with this request.
ClientConnectedId	The ID of the end client connection associated with this request.
ControlNo	A unique control ID used by the GateKeeper server.
RealmId	The Realm ID of the user associated with this request.

GroupId	The Group ID of the user associated with this request.
HostAddr	The address of the originating host
RequestType	The type of this request (Open, Create or Stage).
OFlag	The Open flags associated with this file open request.
StageFlags	Flags associated with this file stage operation.
StageLength	The number of bytes to stage.
StageOffset	The offset of the file where the stage is to begin.
StageStorageLevel	The Storage Class level that the file will be staged to.
UserId	The ID of the user associated with this request.
WaitTime	The length of time this request has been waiting.

F.102.8.1. Action Filters

Every RTM record has an action or operation you can filter by. The most common actions are given below:

Table 47. Common Action Filters

Name	**Description**
open	The RTM entries with an "open" operation.
read	The RTM entries with a "read" operation.
write	The RTM entries with a "write" operation.
migrate	The RTM entries with a "migrate" operation.
stage	The RTM entries with a "stage" operation.
create	The RTM entries with a "create" operation.
purge	The RTM entries with a "purge" operation.
unlink	The RTM entries with a "unlink" operation.
Move tape segment	The RTM entries with a "Move tape segment" operation.
Tape handoff	The RTM entries with a "Tape handoff" operation.
verify	The RTM entries with a "verify" operation.
changecos	The RTM entries with a "changecos" operation.
force migrate	The RTM entries with a "force migrate" operation.
stage batch	The RTM entries with a "stage batch" operation.

F.103. schema_registrar(7)

F.103.1. Name

schema_registrar - Register an XML Schema with the database

F.103.2. Syntax

schema_registrar -f *xml_schema_file* -a *xml_schema_alias*
[-c *schema name*] [-s *subsys_db_name*] [-h]

F.103.3. Description

schema_registrar is a utility that provides a method for registering an XML Schema with the database. Registering an XML schema file with the database allows it to be used to validate incoming updates to the User-defined Attributes.

schema_registrar should be run as hpssdb.

F.103.4. Options

-f *xml_schema_file*

XML Schema Definition File (xsd)

-a *xml_schema_alias*

XML Schema Alias

-s *subsys_db_name*

Subsystem Database (default: subsys1)

-c *schema_name*

HPSS Schema Name (default: hpss)

-h

Usage Help

F.103.5. Examples

1. This example registers the default HPSS XML schema.

```
$ schema_registrar -f hpss_schema.xsd -a "default_schema"
DB20000I  The REGISTER XMLSCHEMA command completed successfully.
DB20000I  The COMPLETE XMLSCHEMA command completed successfully.
```

F.104. scrub(7)

F.104.1. Name

scrub - Interactive interfaces to HPSS Client APIs

F.104.2. Syntax

scrub

F.104.3. Description

This utility provides an interactive, command-line test interface in using and exercising the HPSS client application programming interfaces (APIs).



Scrub is a test tool used by developers for testing the client api, core server, and other areas of HPSS. As such, it does not provide many of the protections offered by other interfaces into the system. For this reason, it must be treated by production sites with caution. It is available for limited use by production sites for information gathering and for other operations described in official HPSS documentation. To prevent unintended consequences, any other use at production sites should be cleared with an HPSS support representative.

If using kerberos authentication, the user should be logged into kerberos prior to invoking scrub. Note that some commands (e.g., "migrate") may require special permissions with the Core Server.

F.104.4. Commands

Use the scrub `help` command. For help on a specific command, enter

```
help <command>
```

F.104.5. Usage

If using kerberos, you must first log in via kerberos and then issue the command scrub.

F.104.6. Example

In the following scrub commands, an HPSS file is created in class-of-service 3. 100 megabytes are written into the file in 8mb chunks using a data pattern that is equal to the byte offset. The last 20mb are then overwritten using a byte pattern of all binary ones followed by writing 50mb of binary zeros. The file is then read back for content verification.

```
% scrub
scrub> hints cos-id 3
scrub> open testfile crw
scrub> write 100mb 8mb
```

```
scrub> lseek -20mb
scrub> write 20mb format=1
scrub> write 50mb format=0
scrub> close
scrub> open testfile r
scrub> read 80mb
scrub> read 20mb 8mb format=1
scrub> read 50mb 4mb format=0
scrub> close
scrub> quit
```

F.104.7. Files

/opt/hpss/bin/scrub

Scrub program

F.105. scsi_home(7)

F.105.1. Name

scsi_home - Display SCSI PVR home locations

F.105.2. Syntax

scsi_home -l *PVR Generic Descriptive name* [-x] [-e] [-h]
[-?] [-g *hcfg*] [-s *subsys*] [-p *utility principal name*]
[-k *utility auth*] [-m *auth mech*]

F.105.3. Description

scsi_home is a utility that provides a method of accessing SCSI Home Locations. This tool is only supported by the SCSI PVR.



This tool must be run as root.

The tool prints out location types for each slot in an abbreviated form; these are:

DTE

Data Transfer Element (Drive)

STE

Storage Element (Cartridge Storage Slot)

IEE

Import Export Element (Door Slot)

TE

Transport Element (Arm)

Some libraries which have multiple robot arms only report a single transport element; users should not expect the transport element to contain cartridge information.

This tool prints out slot flags for various slot types; these are:

For IEE slots only:

I - Import-capable

E - Export-capable

C - Connected Medium Changer

A - Accessible

For all slots: R - Reserved (Slot is being used for some movement)

F.105.4. Options

-l *PVR name*

The descriptive name of the HPSS PVR which manages the tapes whose home location is to be modified. This option must be specified.

-x

Show import slots; full import slot will appear.

-e

Show empty slots; slots will appear even if they have no cartridge. Does not cause import slots to appear.

-h

Show tool help.

-?

Show tool help.

-g *Global Name*

The global configuration database name. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

-s *Schema Name*

The schema database Name. If set, the environment variable HPSS_MM_SCHEMA_NAME will be used if an argument is not specified on the command line.

-p *Principal Name*

The server principal name which this utility will run under. The default value is picked up from the environment variable HPSS_PRINCIPAL_SSM.

-k *Authenticator*

The authenticator which this utility will run with. The default value is picked up from the environment variable HPSS_PRIMARY_AUTHENTICATOR.

-m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

F.105.5. Examples

This example prints out cartridge home locations for drives and storage elements.

```
     $/opt/hpss/bin/scsi_home -l "SCSI PVR"

Library Physical Volumes:

    Location    Home Location   Cartridge   Flags
```

```
         260        4155      100101

        4096        4096      000074

        4097        4097      AD9053

        4098        4098      000086

        4099        4099      100148
            .
            .
            .

    Flag values:    I=Import E=Export R=Reserved
            A=Accessible C=Connected Medium Changer
```

This example prints out cartridge home locations for drives, import/export, and storage elements.

```
     $/opt/hpss/bin/scsi_home -l "SCSI PVR" -x

 Library Physical Volumes:

    Location     Type   Home Location   Cartridge   Flags

         260      DTE       4155      100101

          16      IEE          0      000100         IEA

        4096      STE       4096      000074

        4097      STE       4097      AD9053

        4098      STE       4098      000086

        4099      STE       4099      100148
            .
            .
            .
        Flag values:    I=Import E=Export R=Reserved
                        A=Accessible C=Connected Medium Changer
```

This example prints out cartridge information for drives, import/export, transport, and storage elements regardless of whether a cartridge is present or not.

```
     $/opt/hpss/bin/scsi_home -l "SCSI PVR" -x -e

 Library Physical Volumes:
```

```
    Location     Type   Home Location   Cartridge   Flags

         256      DTE          0

         257      DTE          0

         258      DTE          0

         259      DTE          0

         260      DTE       4155      100101

         261      DTE          0

         16   IEE          0      000100    IEA

         17   IEE          0                IEA
              .
              .
              .
       Flag values:    I=Import E=Export R=Reserved
                       A=Accessible C=Connected Medium Changer
```

F.106. shelf_tape(7)

F.106.1. Name

shelf_tape - HPSS Shelf Tape Utility

F.106.2. Syntax

shelf_tape -r *PVR Descriptive Name*
-f *Input File Name* [-c]

shelf_tape
-o {*age* | *access* | *count*}
-d *Minimum Days Since Last Mount*
-n *Number Of Tapes To Be Shelved*
[-q *Storage Class Id*]
[-g *HPSS Global Database Name*]
[-s *Database Schema Name*]
[-p *Utility Principal Name*]
[-k *Authenticator*]
[-m *Authentication Mechanism*]
[-a]

F.106.3. Description

"shelf_tape" is a HPSS System Maintenance Utility. It is used by an HPSS system administrator to select and move tapes from a tape library to an off-line storage area; this is call "Tape Checkout Requests". It can also be used to move tapes from off-line storage to a tape library; this is called "Tape Checkin Requests". When a tape has been "checked out", it is also referred to as "shelved".

F.106.3.1. Tape Checkout Requests

For tape checkout requests, an administrator can either provide the selection criteria and let the shelf_tape utility select the candidate tapes, or the administrator can specify the tapes to be shelved. In either case, a tape has to be DISMOUNTED and in EOM or RETIRED state before it can be selected or shelved from a tape library.

If the selection is done by the shelf_tape utility, -r, -o, -d and -n parameters are required. If the selection shall be confined in a single Storage Class, then -q is required.

F.106.3.2. Tape Checkin Requests

For tape checkin requests, the administrator must specify the -c option along with a list of the tapes to be checked in (-f option). A tape has to be in the CHECKED OUT state in order to be a candidate for checkin.

F.106.4. Options

-r *PVR Name*

The descriptive name of the HPSS PVR which manages the tapes to be shelved. This option must be specified.

-a

An advisory mode option. This option may be used to display the selected tapes on the screen without moving the tapes to/from the tape library. The advisory mode can be used to verify which tapes can be shelved when the tapes are provided in the -f option.

-o age | access | count

If "age" is specified, then the selection criteria is based on the age of the data. If "access" is specified, then the selection criteria is based on the frequency of data access. If "count" is specified, then the selection criteria is based on the number of accesses to the tape. This option is only valid for Tape Checkout Requests and can not be used with the -f option. This option must be used with the -d and -n options.

-d *Minimum Access Days*

This option precludes the tape that has been accessed in the last specified days. This option is only valid for Tape Checkout Requests and can not be used with the -f option. This option must be used with the -o and -n options.

-n *Number*

The number of tapes to be shelved. This option is only valid for Tape Checkout Requests and can not be used with the -f option. This option must be used with the -o and -d options.

-q *Storage Class*

The tape should be selected from this Storage Class only. If not specified, the selection is from all the Storage Classes. This option is only valid for Tape Checkout Requests and can not be used with the -f option.

-f *Tape List*

The pathname to a file which contains the labels of the tapes to be shelved or checked in. This option must be used for Tape Checkin Requests. This option is optional for Tape Checkout Requests; but can not be used with the -o, -d, -n and -q options.

-c

Check in the requested tapes. This option must be used for Tape Checkin Requests.

-g *Global Name*

The global configuration database Name. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

-s *Schema Name*

The schema database Name. If set, the environment variable HPSS_MM_SCHEMA_NAME will be used if an argument is not specified on the command lin

-p *Principal Name*

The server principal name which this utility will run under. The default value is picked up from

the environment variable HPSS_PRINCIPAL_SSM.

-k *Authenticator*

The authenticator which this utility will run with. The default value is picked up from the environment variable HPSS_PRIMARY_AUTHENTICATOR.

-m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

F.106.5. Example

All the examples except the last one assume the HPSS Generic Server Configuration Path, Keytab Path, Security Mechanism, and Utility Principal Name have been properly set. These environment variables are usually configured in the /var/hpss/etc/env.conf file.

Shelve the 10 tapes which hold the files that were created the earliest and the tapes that have not been accessed in the past 5 days.

```
shelf_tape -r "STK PVR" -o age -d 5 -n 10
```

Identify the 10 tapes which hold the files that were created the earliest and the tapes that have not been accessed in the past 5 days.

```
shelf_tape -r "STK PVR" -o age -d 5 -n 10 -a
```

Shelve the 10 tapes which hold the files that were created the earliest and the tapes that have not been accessed in the past 5 days. These tapes should be selected from Storage Class 2 only.

```
shelf_tape -r "STK PVR" -o age -d 5 -n 10 -q 2
```

Shelve the 10 tapes which have not been accessed for the longest time.

```
shelf_tape -r "STK PVR" -o access -d 0 -n 10
```

Shelve the 10 tapes which have not been accessed for the longest time, but exclude the tapes which have been accessed in the last 2 days. (Note: If all the tapes have been accessed in the last 2 days, no tapes will be shelved.)

```
shelf_tape -r "STK PVR" -o access -d 2 -n 10
```

Shelve the tapes specified in the /dir1/file1 file.

```
      shelf_tape -r "STK PVR" -f "/dir1/file1"
```

```
(Note: the contents of /dir1/file1 are as following
       cat /dir1/file1
```

```
   A00001
   A00002
   A00003
   A00004
)
```

Shelve the 10 tapes which hold the files that were created the earliest and the tapes that have not been accessed in the past 5 days. (Note: This example assumes none of the required environment variables have been set.)

```
      shelf_tape -g cfg -s hpss -m krb
        -p hpssssm -k /var/hpss/etc/hpss.keytab -r "STK PVR" -o age -d 5 -n 10
```

Check in the tapes specified in the /dir1/file1 file.

```
      shelf_tape -r "STK PVR" -f /dir1/file1 -c
```

F.106.6. See Also

shelf_tape_util

F.107. shelf_tape_util(7)

F.107.1. Name

shelf_tape_util - HPSS Shelf Tape Utility

F.107.2. Syntax

shelf_tape_util [*OPTIONS*] -t *Tape_Label* -o *M* | *S*

shelf_tape_util [*OPTIONS*] -r *PVR_desc_name* -o *L*

F.107.3. Description

shelf_tape_util is a HPSS System Maintenance Utility. It is used by a HPSS system administrator to either list all the pending check in tapes, manually mount a referenced shelved tape when the tape robot is down, or update the cartridge state in the cartridge metadata file.



The Manually Mount feature should only be used when the tape robot is not functioning and the referenced tape is on shelf. After the manually mounted tape has been manually removed, the admin has to run the Update Cartridge State feature in order to reset the cartridge state to SHELVED.

F.107.4. Options

-g *Global Name*

The global configuration database Name. file. If set, the environment variable HPSS_GLOBAL_DB_NAME will be used if an argument is not specified on the command line.

-s *Schema Name*

The schema database Name. file. If set, the environment variable HPSS_MM_SCHEMA_NAME will be used if an argument is not specified on the command line.

-p *Principal Name*

The server principal name which this utility will run under. The default value is picked up from the environment variable HPSS_PRINCIPAL_SSM.

-k *Authenticator*

The authenticator which this utility will run with. The default value is picked up from the environment variable HPSS_PRIMARY_AUTHENTICATOR.

-m *Mechanism*

The security mechanism used to authenticate the client. If not provided in the command line, the value of the environment variable HPSS_PRIMARY_AUTHN_MECH will be used.

-o *M*|*L*|*S*

[M]ount a referenced shelved tape, [L]ist all the pending check in tapes, [S]helf to change the cartridge state to SHELVED.

-t

Tape label (for -o M or -o S). Note that this is the 6-character cartridge label (e.g. A00100).

-r

The descriptive name of the PVR for which to list pending checkins (for -o L).

F.107.5. Example

Both examples assume the HPSS Generic Server Configuration DB Name, Authenticator, Authentication Mechanism, and Utility Principal Name have been properly set. These environment variables are usually configured in the /var/hpss/etc/env.conf file.

After manually mounting the referenced shelved tape A00001, enter

```
        shelf_tape_util -o M -t A00001
```

Change the cartridge state to SHELVED for tape A00001, enter

```
        shelf_tape_util -o S -t A00001
```

For listing all the pending check in tapes, enter

```
        shelf_tape_util -r "STK PVR" -o L
```

F.107.6. See Also

shelf_tape

F.108. site.conf(7)

F.108.1. Name

site.conf - HPSS Site Configuration File

F.108.2. Description

The site.conf file provides a structured file containing configuration information describing the site and its realms.

F.108.3. Location

The search path for the site.conf file uses the following algorithm:

1. The path specified by the HPSS_SEC_SITE_CONF environment variable
2. ~hpss/etc/site.conf

F.108.4. Syntax

The syntax of entries in the site.conf file areas follows that of HPSS.conf, but must contain the following stanza:

```
HPSS Site Configuration = {
   Site Name = <site name>
   <Global Settings>
   Realms = {
     <realm name> = {
       Mechanism = <mechanism>
       <mechanism-specific options...>
     }
     <additional realms...>
   }
}
```

F.108.4.1. Site Name

For servers, this is the name of the site. When endpoints are created, they include a site name for clients which want to connect to a specific site.

For clients, this is the name of the default site to connect to. It is possible to connect to any site whose endpoints are described in ep.conf, but if the destination is not overridden, the client API will connect to this site.

F.108.4.2. Global Settings

KEX Type

The key exchange type. Optional. Possible values can be listed with hpss_list_enc kex. Default is

Diffie-Hellman finite field 2048 bits.

Cipher Type

The cipher used in UNIX login server communication. Optional. Possible values can be listed with hpss_list_enc enc. Default is 256-bit AES in CBC mode.

Digest Type

The base algorithm used in signing UNIX login server communication. Optional. Possible values can be listed with hpss_list_enc digest. Default is SHA-256.

UNIX Server Certificate

The certificate that the server presents to the client.

UNIX Server Key

The private key for the server certificate.

UNIX CA File

The file used as the certificate authority for the server certificate.

UNIX CA Dir

The directory which contains certificate authorities.

UNIX Server Address

The address that clients will connect to for UNIX logins.

UNIX Server Port

The port used by the UNIX login service.

Mover Cipher Type

The encryption cipher used in control communications with the movers. Optional. If not specified, communication will not be encrypted. The *hpss_list_enc enc* command can provide supported types.

F.108.4.3. Realm definition

A stanza definition is named after the realm that it defines. Its first entry must be the mechanism type. The additional required and allowed items depend on the mechanism type. Multiple realms may be listed. In this case, the first realm will be the primary realm while the following realms will be considered "trusted realms". These are realms which are allowed to identify users in the same HPSS system. Note that mkhpss will only set up one realm, the primary realm.

Defined realms are used by clients and servers alike to translate between users and their user/group IDs, and to determine group membership. Clients may not need this information, if they are simple enough, i.e. only sending a file to an authenticated user's directory. However, the client API at present requires at least the primary realm to be defined.

Directory

The Directory Mechanism may be either "ldap" or "unix". This specifies whether accounts are created using UNIX style passwd/group files or in an HPSS-specific LDAP directory. Required.

Auth Type

The authentication mechanism to use, either "unix" or "krb5". This specifies how users are authenticated. Required.

Realm ID

The realm identifier number. Required.

F.108.4.3.1. The UNIX Mechanism

PAM Stack

The name of the PAM stack that ultimately authenticates and logs in UNIX users. Required.

Use System UNIX [Authentication|Directory]

If present, this tells HPSS whether to query the system for user accounts. If absent, HPSS will use its own lookup mechanism, referred to as HPSS UNIX. Use of NSS or local system accounts requires this setting to be present. If present for Authentication, then only authentication will be system dependent. If present for Directory, then only HPSS passwd/group files will be consulted for directory information.

Passwd

Set the path to the passwd account file. Required for HPSS UNIX Directory only.

Group

Set the path to the group account file. Required for HPSS UNIX Directory only.

Shadow

Set the path to the shadow password file. Required for UNIX Authentication only. This file supports keytab authentication regardless of UNIX system or UNIX HPSS authentication.

F.108.4.3.2. The LDAP Mechanism

This mechanism stores HPSS accounts directly in a LDAP server using a proprietary HPSS schema. This mechanism is not intended for NSS LDAP. For NSS, the system should be configured at the system level for LDAP, the **UNIX** mechanism should be used, and **Use System UNIX** should be present.

URL

Sets the connection URL for LDAP. Required.

Binding Type

Sets the type of binding that will be done. Required. Possible settings are:

NONE

Perform anonymous binding. This implies that the database allows anybody to query and update it.

SIMPLE

Perform simple DN and password binding. Also requires a **Stash File** to be specified.

GSSAPI

Perform GSSAPI binding. HPSS servers will use their Kerberos principals when binding. Clients which intend to query LDAP must have appropriate ACLs granted in the LDAP database.

EXTERNAL

Perform binding from an external source. For secure LDAP connections, the external binding comes from the client certificate's subject name. Therefore, external binding with secure LDAP requires a **Client Certificate** and a **Client Key.** For UNIX sockets (ldapi://), the binding DN is constructed from the system UID and GID of the process that is binding. For insecure LDAP connections, this binding method cannot be used.

CA Path is Directory

If present, the given path for trusted certificate authorities is a directory. OpenSSL allows for directories which contain multiple file keys named after their hash. In addition, versions on RedHat can use the NSS database method which also resides in a directory.

CA Path

Set to the path for trusted certificate authorities. Required for secure LDAP connections. Connecting clients will use the certificates in this path to verify the LDAP server's certificate. The server will use this path to verify the client certificates.

Client Certificate

Set to the path for the client certificate. Required for secure LDAP connections when *EXTERNAL* binding is used. Optional for secure LDAP with other binding types. If used, **Client Key** must also be provided.

Client Key

Set to the path for the client certificate key. Used when and only when a client certificate is provided.

Stash File

Set to the path for the HPSS LDAP stash file which contains DNs and passwords. Required for *SIMPLE* binding.

Debug LDAP

Turns on debug output for LDAP connections if present.

Debug BER

Turns on debug output for the encoding of LDAP communications if present.

Debug File

Set to the path of a file to write LDAP debug output. Without this setting, debug output is sent to STDERR.

F.108.5. Example

```
HPSS Site Configuration = {
  Site Name = site.example.com
  UNIX CA File = /var/hpss/etc/login_service_cert.pem
  UNIX Server Certificate = /var/hpss/etc/login_service_cert.pem
  UNIX Server Key = /var/hpss/etc/login_service_cert.key
  UNIX Server Address = hpss-system.example.com
  UNIX Server Port = 1024
  Realms = {
    realm1.example.com = {
      Directory = unix
      Auth Type = unix
      Shadow = /var/hpss/etc/shadow
      Use System UNIX
      PAM Stack = hpss
      Realm ID = 10000
    }
    realm2.example.com = {
      Directory = ldap
      Auth Type = unix
      PAM Stack = hpss
      URL = ldaps://ldap.example.com/cn=example.com
      Binding Type = SIMPLE
      Stash File = /var/hpss/etc/ldap_stash
      CA Path is Directory
      CA Path = /etc/cacerts
      Shadow = /var/hpss/etc/shadow2
    }
  }
}
```

F.108.6. See Also

HPSS.conf

HPSS User's Guide

HPSS Admin Guide

F.108.7. Author

Daniel Sands, Sandia National Labs, Albuquerque, NM

F.109. ssmapi_token(7)

F.109.1. Name

ssmapi_token - List, add, and remove SSMAPI tokens

F.109.2. Synopsis

ssmapi_token [-h] [--list] [--remove *CLIENT_ID*] [--add *USER*] [--days *COUNT*]

Either --list or --remove must be specified.

F.109.3. Description

This tool enables HPSS administrators to list, add, and remove SSMAPI web tokens.

F.109.4. Required Arguments

One of these is required:

-l,--list
: Lists all of the tokens, including the user who created the token, the granted roles, the expiration date, and client ID.

-r,--remove *CLIENT_ID*
: Remove the specified *CLIENT_ID*. The *CLIENT_ID* can be found using the list option.

-a,--add *USER*
: Add the specified *USER*. The *USER* must be a valid HPSS user.

F.109.5. Optional Arguments

-h, --help
: show a help message and exit

--days *COUNT*
: Number of days until token expires (default: 90, max: 90)

F.109.6. Prerequisites

The following software prerequisites are required for the ssmapi_token tool:

- Python 3.6 or later

F.110. xmladdindex(7)

F.110.1. Name

xmladdindex - Add an XML index to the User-defined Attributes table

F.110.2. Syntax

xmladdindex -a *index_alias* -p *xml_pattern* -t *pattern_type*
[-s *subsys_db_name*] [-c *db_schema*] [-h]

F.110.3. Description

xmladdindex is a utility that provides a method for adding an XML index to the User-defined Attributes table.

xmladdindex should be run as root with the db2profile sourced, and used to create xml indexes for User-defined Attributes which the administrator determines are good candidates for indexing. Reasons for this would include that the site would like to search for objects which contain the indexed attribute.

Note that Db2 support a varchar with a fixed length, but HPSS does not currently support UDAs with this index type as it can lead to poor index behavior depending upon the UDA content.

F.110.4. Options

-a *index_alias*

XML Index Alias

-p *xml_pattern*

XML Pattern

-t *pattern_type*

XML Pattern Type (double, varchar hashed, date timestamp)

-s *subsys_db_name*

Subsystem Database (default: subsys1)

-c *db_schema*

Database Schema (default: hpss)

-h

Usage Help

F.110.5. Examples

This example adds an XML index "/hpss/author" as "author_index"

```
$ xmladdindex -a author_index -p /hpss/author -t "varchar hashed"
  DB20000I  The SQL command completed successfully.
```

This example adds an XML index "/hpss/expire" as "expire_date"

```
$ xmladdindex -a expire_date -p /hpss/expire -t date
  DB20000I  The SQL command completed successfully.
```

F.110.6. See Also

xmldeleteindex

F.111. xmldeleteindex(7)

F.111.1. Name

xmldeleteindex - Remove an XML index from the User-defined Attributes table

F.111.2. Syntax

xmldeleteindex -a *xml_index_alias* [-s *subsys_db_name*]
[-c *db_schema*] [-h]

F.111.3. Description

xmldeleteindex is a utility that provides a method for deleting an XML index from the User-defined Attributes table.

xmldeleteindex should be run as root with the db2profile sourced, and used to delete xml indexes from the User-defined Attributes table.

F.111.4. Options

-a *xml_index_alias*

XML Index Alias

-s *subsys_db_name*

Subsystem Database Name (default: subsys1)

-c *db_schema*

Schema Name (default: hpss)

-h Usage Help

F.111.5. Examples

This example deletes an XML index aliased as "author_index"

```
$ xmldeleteindex -a author_index
  DB20000I  The SQL command completed successfully.
```

F.111.6. See Also

xmladdindex