

# HPSS Admin Guide

February 06, 2025: High Performance Storage System 11.2

# Table of Contents

1. Release 11 .....	3
1.1. New features in 11.2 .....	3
1.1.1. Client Striped IOD .....	3
1.1.2. LibAIO Support .....	3
1.1.3. S3 Kerberos Support .....	3
1.2. Features retired in 11.2 .....	4
1.3. Features deprecated in 11.2 .....	4
1.4. Changes to existing HPSS features in 11.2 .....	4
1.5. API changes in HPSS 11.2 .....	4
1.5.1. Simplified PIO Interface .....	4
1.5.2. Client Striped IOD in PIO .....	5
1.6. New features in 11.1 .....	5
1.6.1. HPSS Recurse tool .....	5
1.6.2. VV Selection Priority (CR 411) .....	5
1.6.3. Purge By Largest File First .....	6
1.6.4. Secondary Data Copies in AWS .....	6
1.7. Features retired in 11.1 .....	7
1.8. Features deprecated in 11.1 .....	7
1.9. Changes to existing HPSS features in 11.1 .....	7
1.9.1. Updates to RTM Summary List window and rtmu tool .....	7
1.10. API changes in HPSS 11.1 .....	7
2. HPSS basics .....	8
2.1. Introduction .....	8
2.2. HPSS capabilities .....	8
2.2.1. Network-centered architecture .....	8
2.2.2. High data transfer rate .....	8
2.2.3. Parallel operation .....	9
2.2.4. Based on standard components .....	9
2.2.5. Data integrity through transaction management .....	9
2.2.6. Multiple hierarchies and Classes of Services .....	9
2.2.7. Storage subsystems .....	10
2.3. HPSS components .....	10
2.3.1. HPSS files, filesets, volumes, storage segments and related metadata .....	12
2.3.2. HPSS servers .....	14
2.3.3. HPSS storage subsystems .....	18
2.3.4. HPSS infrastructure .....	18
2.3.5. HPSS user interfaces .....	20
2.3.6. HPSS management interfaces .....	21

2.3.7. HPSS policy modules .....	21
2.4. HPSS hardware platforms .....	23
2.4.1. Server platforms .....	23
2.4.2. Client platforms .....	23
2.4.3. Mover platforms .....	24
3. HPSS planning .....	25
3.1. Overview .....	25
3.1.1. HPSS system architecture .....	25
3.1.2. HPSS configuration planning .....	26
3.1.3. Purchasing hardware and software .....	28
3.1.4. HPSS operational planning .....	28
3.1.5. HPSS deployment planning .....	29
3.2. Requirements and intended uses for HPSS .....	29
3.2.1. Storage system capacity .....	29
3.2.2. Required throughputs .....	30
3.2.3. Load characterization .....	30
3.2.4. Usage trends .....	30
3.2.5. Duplicate file policy .....	30
3.2.6. Charging policy .....	30
3.2.7. Security .....	31
3.2.7.1. Cross realm access .....	31
3.2.8. HPSS availability options .....	32
3.3. Prerequisite software considerations .....	32
3.3.1. Prerequisite software overview .....	32
3.3.1.1. DB2 .....	33
3.3.1.2. OpenSSL .....	33
3.3.1.3. Kerberos .....	33
3.3.1.4. LDAP and IBM Kerberos .....	33
3.3.1.5. Java .....	33
3.3.1.6. Use of libTI-RPC .....	34
3.3.1.7. Jansson .....	34
3.3.1.8. STK Tools .....	34
3.3.1.9. Apache Commons Codecs .....	34
3.3.2. Prerequisite summary By HPSS node type .....	34
3.3.2.1. HPSS server nodes .....	34
3.3.2.2. HPSS client nodes .....	35
3.4. Hardware considerations .....	35
3.4.1. Network considerations .....	35
3.4.2. Robotically mounted tape .....	36
3.4.2.1. Drive-controlled LTO libraries (IBM, Spectralogic) .....	36
3.4.2.2. Oracle StorageTek .....	37

3.4.2.3. Oracle StorageTek tape libraries that support ACSLS . . . . .	37
3.4.3. Manually mounted tape . . . . .	37
3.4.4. Tape devices . . . . .	37
3.4.4.1. Multiple media support . . . . .	37
3.4.5. Disk devices . . . . .	40
3.4.6. AWS Tape Gateway . . . . .	41
3.4.7. Special bid considerations . . . . .	41
3.5. HPSS sizing considerations . . . . .	41
3.5.1. HPSS user storage space . . . . .	42
3.5.2. HPSS infrastructure storage space . . . . .	42
3.5.2.1. HPSS and DB2 file systems . . . . .	45
3.5.2.2. HPSS metadata space . . . . .	48
3.5.2.3. HPSS file systems . . . . .	50
3.5.3. System memory and disk space . . . . .	52
3.5.3.1. Operating system disk spaces . . . . .	52
3.5.3.2. System disk space requirements for running SSM . . . . .	52
3.5.3.3. System memory and paging space requirements . . . . .	52
3.6. HPSS interface considerations . . . . .	53
3.6.1. Client API . . . . .	53
3.6.2. FTP . . . . .	54
3.6.3. Parallel FTP . . . . .	54
3.7. HPSS server considerations . . . . .	54
3.7.1. Core Server . . . . .	55
3.7.2. Migration/Purge Server . . . . .	57
3.7.3. Gatekeeper . . . . .	60
3.7.4. PVL . . . . .	61
3.7.5. PVR . . . . .	62
3.7.5.1. STK PVR . . . . .	62
3.7.5.2. Operator PVR . . . . .	63
3.7.5.3. SCSI PVR . . . . .	63
3.7.6. Mover . . . . .	63
3.7.6.1. Tape devices . . . . .	63
3.7.6.2. Disk devices . . . . .	63
3.7.6.3. Performance . . . . .	64
3.7.7. Logging service . . . . .	65
3.7.8. Startup Daemon . . . . .	65
3.7.9. Storage System Management . . . . .	65
3.8. Storage subsystem considerations . . . . .	67
3.9. Storage policy considerations . . . . .	68
3.9.1. Migration policy . . . . .	68
3.9.1.1. Migration policy for disk . . . . .	68

3.9.1.2. Migration policy for tape	68
3.9.2. Purge policy	69
3.9.3. Accounting policy and validation	70
3.9.4. Security policy	71
3.9.4.1. Client API	72
3.9.4.2. FTP/PFTP	72
3.9.4.3. Name space	72
3.9.4.4. Security audit	72
3.9.5. Logging policy	72
3.9.6. Gatekeeping	73
3.10. Storage characteristics considerations	75
3.10.1. Storage class	76
3.10.1.1. Media block size selection	77
3.10.1.2. Virtual volume block size selection (disk)	77
3.10.1.3. Virtual volume block size selection (tape)	77
3.10.1.4. Stripe width selection	77
3.10.1.5. Blocks between tape marks selection (tape only)	79
3.10.1.6. Minimum storage segment size selection (disk only)	79
3.10.1.7. Maximum storage segment size selection	80
3.10.1.8. Maximum VVs to write (tape only)	81
3.10.1.9. Average number of storage segments (disk only)	81
3.10.1.10. PV estimated size and PV size selection	81
3.10.1.11. Optimum access size selection	81
3.10.1.12. Some recommended parameter values for supported storage media	82
3.10.2. Storage hierarchy	85
3.10.3. Class of Service	85
3.10.3.1. Selecting minimum file size	86
3.10.3.2. Selecting maximum file size	86
3.10.3.3. Selecting stage code	86
3.10.3.4. Selecting optimum access size	87
3.10.3.5. Selecting average latency	87
3.10.3.6. Selecting transfer rate	88
3.10.3.7. StripeLength and StripeWidth hints	88
3.10.4. File families	88
3.11. HPSS performance considerations	89
3.11.1. DB2	89
3.11.2. Bypassing potential bottlenecks	89
3.11.3. Configuration	90
3.11.4. FTP/PFTP	90
3.11.5. Client API	91
3.11.6. Core Server	92

3.11.7. Logging	92
3.11.8. Cross-realm trust	92
3.11.9. Gatekeeping	92
3.11.10. HPSSFS-FUSE interface	93
3.12. HPSS metadata backup considerations	93
3.13. HPSS security considerations	94
4. System preparation	95
4.1. General setup	95
4.2. Set up file systems	96
4.2.1. DB2 file system	96
4.2.2. HPSS file system	97
4.3. Set up tape libraries	97
4.3.1. Oracle StorageTek	97
4.3.2. SCSI	98
4.4. Verify tape drives	99
4.4.1. Linux	99
4.5. Set up disk drives	100
4.5.1. Linux	100
4.6. Set up network parameters	100
4.6.1. <code>HPSS.conf</code> configuration file	103
4.7. Port mapping and firewall considerations	103
4.8. Semaphore values	104
4.9. Enable Core Dumps	106
5. HPSS installation and infrastructure configuration	108
5.1. Prepare for installation	108
5.1.1. Distribution media	108
5.1.2. Software installation packages	108
5.1.3. Create owner account for HPSS files	109
5.1.4. Installation target directory preparation	109
5.2. Install prerequisite software	109
5.2.1. Install Java	109
5.2.2. Install Jansson	110
5.2.3. Install TI-RPC	110
5.2.4. Install Ncurses	110
5.2.5. Install MIT Kerberos	110
5.2.6. Install LDAP (if using LDAP authorization)	110
5.2.7. Install DB2 and set up permanent license	111
5.3. Install HPSS with RPMs	111
5.4. Install HPSS	113
5.4.1. On core	113
5.4.2. On Mover	113

5.4.3. On client	114
5.4.4. On remote PVR	114
5.4.5. Generate and bind the DB2 helper program	115
5.4.6. Update default DB2 link	115
5.5. Configure HPSS infrastructure	115
5.5.1. Navigating and general <b>mkhps</b> behavior	116
5.5.2. Configure HPSS - root subsystem machine	116
5.5.2.1. Pre-installation configuration	117
5.5.2.2. Configure HPSS security services	118
5.5.2.3. Configure DB2 services	130
5.5.2.4. <i>Setting up off-node DB2</i>	137
5.5.2.5. Configure other services	144
5.5.2.6. Create configuration bundle	146
5.5.3. Configure HPSS - secondary subsystem machine	147
5.5.4. Troubleshooting <b>mkhps</b>	147
5.6. Prepare post-installation procedures	148
5.7. Locate HPSS documentation and set up manual pages	149
5.7.1. Documentation and SSM help package	149
5.7.2. Manual pages setup	150
5.8. Define HPSS environment variables	150
5.9. Set up a remote PVR	151
5.10. Tune DB2	153
5.11. Supporting both UNIX and Kerberos authentication for SSM	154
5.12. HPSS IPv6 support	156
5.12.1. Usage examples	157
6. Installation and configuration of the Elastic (ELK) Stack	160
6.1. Installing the ELK stack	160
6.1.1. Install Filebeat	160
6.1.2. Install Logstash	161
6.1.3. Install Elastic	161
6.1.4. Install Kibana	162
6.1.5. Scaling Elastic	162
6.2. Installing the HPSS data capture components	162
6.2.1. HPSS data capture scripts	162
6.2.2. How to configure the data capture scripts	162
6.3. Setup the HPSS dashboards	163
6.3.1. Load the HPSS templates into Kibana	163
6.3.2. The HPSS dashboards	163
6.4. Captured data	164
7. HPSS S3 interface	167
7.1. Overview	167

7.2. HPSS S3 Interface Setup .....	168
7.3. Interoperation .....	172
7.4. HPSS Specific Options .....	173
7.5. S3 Object Listing .....	174
7.6. S3 Virtual Hosting .....	174
7.7. S3 Buckets .....	174
7.8. S3 Client Setup .....	175
7.8.1. General .....	175
7.8.2. s3cmd .....	175
7.8.3. boto3 .....	175
7.9. Unsupported Operations .....	176
7.10. Limitations .....	177
7.11. Performance .....	178
7.12. Scaling and Load Balancing .....	178
7.13. Setting up for Kerberos use .....	179
8. HPSS configuration overview .....	181
8.1. Introduction .....	181
8.2. Starting the SSM GUI for the first time .....	181
8.3. HPSS configuration roadmap (new HPSS sites) .....	182
8.4. Initial HPSS startup roadmap (all sites) .....	182
8.5. Additional configuration roadmap (all sites) .....	183
8.6. Verification checklists (all sites) .....	183
8.6.1. Configuration checklists .....	184
8.6.2. Operational checklists .....	185
8.6.3. Performance checklist .....	186
9. Security and system access .....	187
9.1. Security services .....	187
9.1.1. Security services configuration .....	187
9.1.2. Security mechanisms .....	188
9.1.2.1. UNIX .....	188
9.1.2.2. UNIX and NIS or LDAP .....	188
9.1.2.3. Kerberos 5 .....	189
9.1.2.4. LDAP .....	189
9.2. HPSS server security ACLs .....	191
9.3. SSM user security .....	193
9.4. Restricting user access to HPSS .....	193
9.4.1. Overview .....	194
9.4.2. Configuration File .....	194
9.4.2.1. File Permissions .....	194
9.4.2.2. File Format .....	194
9.4.3. ra_verify_config .....	195



9.4.4. Refresh Restricted Access Table .....	195
9.4.5. Restriction Types .....	195
9.4.6. Example Restrict Access File .....	195
10. Using SSM .....	197
10.1. The SSM System Manager .....	197
10.1.1. Starting the SSM System Manager .....	197
10.1.2. Tuning the System Manager RPC thread pool and request queue sizes .....	197
10.1.3. Labeling the System Manager RPC program number .....	199
10.2. Quick startup of <b>hpssgui</b> .....	199
10.3. Configuration and startup of <b>hpssgui</b> and <b>hpssadm</b> .....	200
10.3.1. Configuring the System Manager authentication for SSM clients .....	201
10.3.2. Creating the SSM user accounts .....	202
10.3.2.1. The <b>hpssuser</b> utility .....	202
10.3.2.2. SSM user authorization .....	202
10.3.2.3. User keytabs .....	204
10.3.3. SSM configuration file .....	205
10.3.3.1. login.conf .....	207
10.3.3.2. krb5.conf (for use with Kerberos authentication only) .....	207
10.3.4. SSM help files (optional) .....	208
10.3.5. SSM desktop client packaging .....	208
10.3.5.1. Automatic SSM client packaging and installation .....	209
10.3.5.2. Manual SSM client packaging and installation .....	209
10.3.6. Using SSM through a firewall .....	211
10.3.6.1. The firewall problem .....	211
10.3.6.2. Solutions for operating through a firewall .....	211
10.3.6.3. Example: Using <b>hpssgui</b> through a firewall .....	212
10.4. Multiple SSM sessions .....	213
10.5. SSM window conventions .....	213
10.6. Common window elements .....	216
10.7. Help menu overview .....	219
10.8. Monitor, Operations, and Configure menus overview .....	219
10.8.1. Monitor menu .....	220
10.8.2. Operations menu .....	221
10.8.3. Configure menu .....	223
10.9. SSM-specific windows .....	225
10.9.1. HPSS Login .....	225
10.9.2. About HPSS .....	227
10.9.3. HPSS Health and Status .....	228
10.9.3.1. SM server connection status indicator .....	230
10.9.3.2. HPSS status .....	230
10.9.3.3. HPSS statistics .....	232

10.9.3.4. Menu tree .....	233
10.9.3.5. File menu .....	233
10.9.3.6. View menu .....	234
10.9.4. SSM information windows .....	234
10.9.4.1. System Manager Statistics window .....	234
10.9.4.2. User Session Information window .....	239
10.10. SSM list preferences .....	241
11. Global and subsystem configuration .....	244
11.1. Global Configuration window .....	244
11.2. Storage subsystems .....	249
11.2.1. Subsystems list window .....	249
11.2.2. Creating a new storage subsystem .....	250
11.2.3. Storage Subsystem Configuration window .....	251
11.2.3.1. Create storage subsystem metadata .....	254
11.2.3.2. Create storage subsystem configuration .....	254
11.2.3.3. Create storage subsystem servers .....	255
11.2.3.4. Assign a Gatekeeper if required .....	255
11.2.3.5. Assign storage resources to the storage subsystem .....	255
11.2.3.6. Create storage subsystem fileset and junction .....	255
11.2.3.7. Migration and purge policy overrides .....	256
11.2.3.8. Storage class threshold overrides .....	256
11.2.4. Modifying a storage subsystem .....	256
11.2.5. Deleting a storage subsystem .....	256
11.3. HPSS trashcans .....	257
11.3.1. The trashcan feature is on .....	258
11.3.2. Trashcan operations for the administrator .....	260
12. HPSS servers .....	262
12.1. Server list .....	262
12.2. Server configuration .....	268
12.2.1. Common server configuration .....	270
12.2.1.1. Basic controls .....	270
12.2.1.2. Execution controls .....	271
12.2.1.3. Interface controls .....	273
12.2.1.4. Security controls .....	276
12.2.1.5. Audit policy .....	278
12.2.2. Core Server-specific configuration .....	280
12.2.2.1. Additional Core Server configuration .....	284
12.2.2.2. Core Server I/O limit configuration .....	286
12.2.2.3. Core Server max open files configuration .....	289
12.2.3. Gatekeeper-specific configuration .....	290
12.2.4. Core Server additional configuration .....	291

12.2.5. Migration/Purge Server (MPS)-specific configuration . . . . .	292
12.2.6. Mover-specific configuration . . . . .	293
12.2.6.1. Mover-specific configuration window . . . . .	293
12.2.6.2. Additional Mover configuration . . . . .	296
12.2.7. Physical Volume Repository (PVR)-specific configuration . . . . .	303
12.2.7.1. Operator PVR-specific configuration window . . . . .	303
12.2.7.2. SCSI PVR-specific configuration window . . . . .	305
12.2.7.3. STK PVR-specific configuration window . . . . .	309
12.2.8. RAIT Engine-specific configuration . . . . .	316
12.2.8.1. RAIT Engine-specific configuration window . . . . .	316
12.2.9. Deleting a server configuration . . . . .	319
12.3. Monitoring server information . . . . .	321
12.3.1. Basic Server Information . . . . .	321
12.3.2. Specific server information . . . . .	324
12.3.2.1. Core Server Information window . . . . .	324
12.3.2.2. Gatekeeper Information window . . . . .	336
12.3.2.3. Migration/Purge Server Information window . . . . .	340
12.3.2.4. Mover Information window . . . . .	340
12.3.2.5. Physical Volume Library (PVL) information window . . . . .	342
12.3.2.6. Operator PVR information window . . . . .	343
12.3.2.7. SCSI PVR information window . . . . .	345
12.3.2.8. STK PVR information window . . . . .	349
12.3.2.9. RAIT Engine Information window . . . . .	352
12.4. Real-Time Monitoring (RTM) . . . . .	354
12.4.1. RTM Summary List . . . . .	354
12.4.2. RTM Detail . . . . .	356
12.5. I/O Abort . . . . .	364
12.5.1. Aborting an I/O . . . . .	364
12.5.2. Expectations in Aborting I/O . . . . .	365
12.6. Starting HPSS . . . . .	366
12.6.1. Starting HPSS prerequisite software . . . . .	366
12.6.2. Starting HPSS servers . . . . .	366
12.6.2.1. Starting the Startup Daemons . . . . .	366
12.6.2.2. Starting SSM . . . . .	367
12.6.2.3. Starting other HPSS servers . . . . .	367
12.6.2.4. Automatic server restart . . . . .	368
12.7. Stopping HPSS . . . . .	368
12.7.1. Shutting down an HPSS server . . . . .	368
12.7.2. Shutting down all HPSS servers . . . . .	369
12.7.3. Halting an HPSS server . . . . .	369
12.7.4. Shutting down the SSM server . . . . .	370

12.7.5. Shutting down the Startup Daemon .....	370
12.7.6. Stopping the prerequisite software .....	370
12.8. Server repair and reinitialization .....	370
12.8.1. Repairing an HPSS server .....	371
12.8.2. Reinitializing a server .....	371
12.9. Forcing an SSM connection .....	375
13. Storage configuration .....	376
13.1. Storage classes .....	376
13.1.1. Configured Storage Classes window .....	376
13.1.2. S3 Polling Intervals .....	378
13.1.3. Disk Storage Class Configuration .....	379
13.1.4. Tape Storage Class Configuration .....	383
13.1.5. Object Store Storage Class Configuration .....	390
13.1.6. Storage class subsystem thresholds .....	395
13.1.6.1. Disk Storage Subsystem-Specific Thresholds .....	395
13.1.6.2. Tape Storage Subsystem-Specific Thresholds .....	398
13.1.7. Changing a storage class definition .....	400
13.1.8. Deleting a storage class definition .....	401
13.2. Storage hierarchies .....	401
13.2.1. Hierarchies window .....	401
13.2.2. Storage Hierarchy Configuration window .....	403
13.2.3. Changing a storage hierarchy definition .....	405
13.2.4. Deleting a storage hierarchy definition .....	405
13.3. Classes of Service .....	405
13.3.1. Classes of Service window .....	405
13.3.2. Class of Service Configuration window .....	407
13.3.3. Changing a Class of Service definition .....	414
13.3.4. Deleting a Class of Service definition .....	415
13.3.5. Changing a file's Class of Service .....	415
13.3.6. Canceling a Class of Service change request .....	416
13.4. Migration policies .....	416
13.4.1. Migration Policies window .....	416
13.4.2. Migration policy configuration .....	418
13.4.2.1. Disk Migration Policy configuration .....	418
13.4.2.2. Tape Migration Policy configuration .....	423
13.4.2.3. Changing a migration policy .....	426
13.4.2.4. Deleting a migration policy .....	426
13.5. Purge policies .....	427
13.5.1. Purge Policies window .....	427
13.5.2. Purge Policy configuration .....	429
13.5.3. Changing a purge policy .....	432

13.5.4. Deleting a purge policy .....	433
13.5.5. Purge parallelism .....	433
13.6. File families .....	433
13.6.1. File Family Configuration .....	434
13.6.2. Changing a file family .....	435
13.6.3. Deleting a file family .....	435
14. Device and drive management .....	436
14.1. Configure a new device and drive .....	436
14.1.1. Devices and Drives window .....	447
14.1.2. Enable variable block sizes for tape devices .....	452
14.1.3. Changing a drive's configuration .....	452
14.1.4. Deleting a drive's configuration .....	453
14.2. Monitoring devices and drives .....	454
14.2.1. Mover Device Information window .....	455
14.2.2. PVL Drive Information window .....	462
14.3. Changing device and drive state .....	467
14.3.1. Unlocking a drive .....	467
14.3.2. Locking a drive .....	467
14.3.3. Repairing the state of a device or drive .....	468
14.3.4. Resetting drive statistics .....	468
14.3.5. Tape Drive Quota list window .....	468
15. Volume and storage management .....	470
15.1. Adding storage space .....	470
15.1.1. Importing volumes into HPSS .....	470
15.1.1.1. Import Tape Volumes window .....	472
15.1.1.2. Selecting import type for tape cartridges .....	477
15.1.1.3. Import Disk Volumes window .....	478
15.1.1.4. Selecting import type for disk volumes .....	483
15.1.1.5. Import Object Store Volumes window .....	483
15.1.2. Creating storage resources .....	487
15.1.2.1. Create Tape Resources window .....	488
15.1.2.2. Prioritize Tape Resource Selection .....	493
15.1.2.3. Aspects of Selection Priority Weights .....	494
15.1.2.4. Create Disk Resources window .....	495
15.1.2.5. Create Object Store Resources window .....	499
15.2. Removing storage space .....	504
15.2.1. Deleting storage resources .....	504
15.2.1.1. Rules for deleting resources .....	504
15.2.1.2. Delete Resources window .....	504
15.2.2. Exporting volumes from HPSS .....	508
15.2.2.1. Rules for exporting volumes .....	508

15.2.2.2. Export Volumes window .....	508
15.3. Monitoring storage space .....	512
15.3.1. Active Storage Classes window .....	512
15.3.2. MPS Disk Storage Class Information .....	517
15.3.3. MPS Tape Storage Class Information .....	524
15.3.4. MPS Object Store Storage Class Information .....	528
15.4. Dealing with a space shortage .....	530
15.4.1. Manually starting migration .....	530
15.4.2. Manually starting purge .....	531
15.4.3. Repacking and reclaiming volumes .....	531
15.4.4. Verifying data integrity .....	533
15.4.4.1. Repack Virtual Volumes window .....	533
15.4.4.2. Reclaim Virtual Volumes window .....	536
15.5. Volume management .....	537
15.5.1. Lookup Cartridges and Volumes window .....	537
15.5.2. PVL Volume Information window .....	538
15.5.3. PVR Cartridge Information window .....	541
15.5.4. Core Server Volume and Segment windows .....	545
15.5.4.1. Core Server Disk Volume information window .....	545
15.5.4.2. Core Server Tape Volume information window .....	551
15.5.4.3. Core Server Object Store Volume information window .....	559
15.5.5. Changing Core Server volume condition .....	564
15.5.6. Moving PVR cartridges to another PVR .....	566
15.5.6.1. Move Cartridges window .....	567
15.6. Monitoring and managing volume mounts .....	570
15.6.1. PVL Job Queue window .....	570
15.6.2. PVL Request Information window .....	574
15.6.3. Canceling queued PVL requests .....	577
15.6.4. Tape Check-In Requests window .....	577
15.6.5. Tape Mount Requests window .....	579
15.6.6. Administrative tape dismounts .....	580
15.7. Storage technology replacement .....	581
16. Logging and status .....	583
16.1. Logging overview .....	583
16.2. Log policies .....	584
16.2.1. Creating a log policy .....	584
16.2.2. Logging Policies window .....	584
16.2.2.1. Logging Policy configuration window .....	585
16.2.3. Changing a log policy .....	588
16.2.4. Deleting a log policy .....	588
16.3. Managing syslog .....	588

16.3.1. Configuring syslog .....	588
16.3.2. Recommendations for syslog configuration .....	589
16.3.3. Interactions between HPSS and syslog .....	589
16.3.4. Configure syslog message size .....	590
16.3.5. Configure multi-part messages .....	590
16.3.6. Restart Syslog After Configuration Change .....	591
16.3.7. Managing rate-limiting and duplication .....	591
16.3.8. Considerations For SELinux .....	592
16.3.9. Managing log archiving .....	592
16.3.10. Recommendations for logrotate configuration .....	594
16.3.11. Viewing HPSS logs .....	595
16.4. Managing SSM alarms and events .....	595
16.4.1. Alarms and Events window .....	596
16.4.2. Alarm/Event Information .....	596
16.4.3. Diagnosing HPSS problems with alarms and events .....	598
16.4.4. Controlling SSM log message handling .....	599
16.4.4.1. Controlling the System Manager log message cache .....	599
16.4.4.2. Controlling log messages displayed by <b>hpssgui</b> and <b>hpssadm</b> .....	600
16.4.5. Media Access Logging .....	601
16.5. Log files of various types - a comprehensive list .....	602
16.6. Disk Mover & VV I/O Metrics .....	607
16.6.1. Sample Log of Disk I/O Operations .....	609
17. Filesets and junctions .....	612
17.1. Filesets & Junctions List .....	612
17.2. Creating an HPSS fileset .....	614
17.2.1. Create Fileset window .....	615
17.3. Managing existing filesets .....	617
17.3.1. Core Server Fileset Information window .....	617
17.4. Deleting filesets .....	620
17.5. Creating a junction .....	621
17.5.1. Create Junction window .....	621
17.6. Deleting a junction .....	622
18. Files, directories and objects by SOID .....	623
18.1. Files and Directories window .....	623
18.1.1. File/Directory Information window .....	624
18.2. Objects by SOID window .....	632
19. Tape aggregation .....	634
19.1. Overview of tape aggregation .....	634
19.2. Tape aggregation performance considerations .....	634
19.3. Ordered Migration and Full Aggregate Recall .....	635
19.3.1. Ordered migration example - read-optimized aggregates .....	636

19.4. Configuring tape aggregation	636
19.5. Caveats	636
20. Tape Ordered Recall (TOR)	638
20.1. Overview of TOR	638
20.2. TOR ordering behavior	638
20.3. TOR performance considerations	639
20.4. Configuring TOR	640
20.5. Caveats	641
21. User accounts and accounting	642
21.1. Managing HPSS users	642
21.1.1. Adding HPSS users	642
21.1.1.1. Add all user ID types	642
21.1.1.2. Add a user ID	643
21.1.1.3. Add an SSM user ID	644
21.1.2. Deleting HPSS users	644
21.1.3. Listing HPSS users	644
21.1.4. Create an SSM client package	645
21.2. Accounting	645
21.2.1. Accounting Policy window	646
21.2.2. Accounting reports and status	648
21.2.2.1. Generating an accounting report	648
21.2.2.2. Accounting Status window	649
21.2.2.3. Interpreting the accounting report	650
21.2.3. Accounting procedures	653
21.2.3.1. Site-defined accounting configuration files and procedures	654
21.2.3.2. Accounting intervals and charges	656
22. Object store accounts and accounting	657
22.1. Managing object store accounts	657
22.1.1. Object Store Accounts window	657
22.1.2. Object Store Account Configuration	658
22.2. Managing object store account identities	658
22.2.1. Object Store Account Identities window	659
22.2.2. Object Store Account Identity Configuration	660
22.3. Managing Object Store Account Buckets	661
22.3.1. Object Store Account Buckets window	661
22.3.2. Object Store Account Bucket Configuration	662
22.4. Managing Object Store Account Mappings	663
22.4.1. Object Store Account Mappings window	663
22.4.2. Object Store Account Mapping Configuration	664
23. User interfaces	666
23.1. Client API configuration	666



23.2. FTP/PFTP daemon configuration .....	668
24. Backup and recovery .....	675
24.1. HPSS metadata backup and recovery .....	675
24.1.1. HPSS administrator responsibilities for DB2 .....	675
24.1.2. Overview of the DB2 backup process .....	676
24.1.2.1. Configuring DB2 for online backup .....	677
24.1.3. Overview of the DB2 recovery process .....	678
24.2. HPSS system environmental backup .....	678
24.2.1. HPSS file system backup .....	678
24.2.2. Operating system backup .....	679
24.2.3. Kerberos backup .....	679
24.3. HPSS user data recovery: Recovering HPSS files from damaged HPSS volumes .....	680
24.4. DB2 monitoring .....	682
24.5. DB2 space shortage .....	682
24.5.1. DMS table spaces .....	682
24.5.2. SMS table spaces .....	683
25. Management tools .....	684
25.1. Utility overview .....	684
25.1.1. Fileset and junction management .....	684
25.1.2. Tape management .....	684
25.1.3. System info .....	685
25.1.4. System management .....	686
25.1.5. User interfaces .....	687
25.1.6. Testing and debugging .....	687
25.1.7. Unsupported tools .....	688
25.2. Tape queue monitoring .....	689
25.2.1. Example .....	689
25.2.2. JSON field descriptions .....	690
25.3. HPSS ACL permission exceptions .....	694
Appendix A: Glossary of terms and acronyms .....	697
Appendix B: References .....	713
Appendix C: Developer acknowledgments .....	715
Appendix D: <b>HPSS.conf</b> configuration file .....	716
PFTP Client Stanza .....	716
PFTP Client Interfaces Stanza .....	722
Multinode Table Stanza .....	727
Network Options Stanza .....	729
PFTP Daemon Stanza .....	735
Transfer Agent Stanza .....	750
Stanzas reserved for future use .....	758
HPSS Environment Variables .....	759

HPSS Environment Variables .....	759
HPSS Misc Environment .....	759
HPSS Server Group Names .....	760
HPSS Server Principal Names .....	760
HPSS Server Principal UIDs .....	761
HPSS Server Executable Names .....	762
SSM Utility locations .....	763
Logging unix files .....	763
Accounting Unix files .....	764
MPS Unix files .....	764
Gatekeeper environment variables .....	764
Database environment variables .....	764
System environment variables .....	765
SSM Specific .....	766
API Specific .....	769
Core Specific .....	771
Logging Specific .....	772
LS Specific .....	773
GHI Specific .....	773
FTPD Specific .....	773
RAIT Engine Specific .....	774
Migration Purge Server Specific .....	774
Mover Specific .....	774
Security Registry & Service Location Specific .....	775
RPC Specific .....	776
SCSI PVR .....	777
RTM Settings .....	777
Installation & Misc .....	778
HPSS Group Names .....	778
HPSS Group Default IDs .....	779
HPSS User Names .....	779
HPSS Server Unix UIDs .....	780
HPSS Object Store Encryption Master Key File .....	780
HPSS Object Store Notifcation Configuration .....	780
Appendix E: The /var/hpss files .....	781

*Copyright notification*

Copyright © 1992-2025 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, AIX, pSeries, and xSeries are trademarks or registered trademarks of International Business Machines Corporation.

AIX and RISC/6000 are 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 preceded 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 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.

## 1.1. New features in 11.2

New features for 11.2 are described below.

### 1.1.1. 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 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.

### 1.1.2. 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.

### 1.1.3. 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.

## 1.2. Features retired in 11.2

None.

## 1.3. Features deprecated in 11.2

None.

## 1.4. 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`.

## 1.5. API changes in HPSS 11.2

### 1.5.1. 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.

## 1.5.2. 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.

# 1.6. New features in 11.1

New features for 11.1 are described below.

## 1.6.1. 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 the *hpss\_recurse* man page for more details.

## 1.6.2. 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, admins 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](#), [hpss\\_update\\_priority\\_by\\_pvr\\_drive](#), and [hpss\\_update\\_priority\\_by\\_create\\_time](#). 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 the `hpss_update_priority_by_pvr_drive`. 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` tool with the `-P` option, or by using the new `hpss_vv_select_priority` script. The `hpss_vv_select_priority` script can be used to create more complex or custom prioritization schemes or to reset VV priorities.

See the [Prioritize Tape Resource Selection](#) section and `man hpss_vv_select_priority` for details.

### 1.6.3. 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.

### 1.6.4. 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.



## 1.7. Features retired in 11.1

HPSS features retired in 11.1 are described below.

## 1.8. Features deprecated in 11.1

HPSS features deprecated in 11.1 are described below.

## 1.9. Changes to existing HPSS features in 11.1

HPSS features significantly altered in 11.1 are described below.

### 1.9.1. 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` command-line tool.

## 1.10. API changes in HPSS 11.1

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

# Chapter 2. HPSS basics

---

## 2.1. 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.

## 2.2. 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.

### 2.2.1. 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*.

### 2.2.2. 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.

### 2.2.3. 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.

### 2.2.4. 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.

### 2.2.5. 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.

### 2.2.6. 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.

## 2.2.7. 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.

## 2.3. 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.

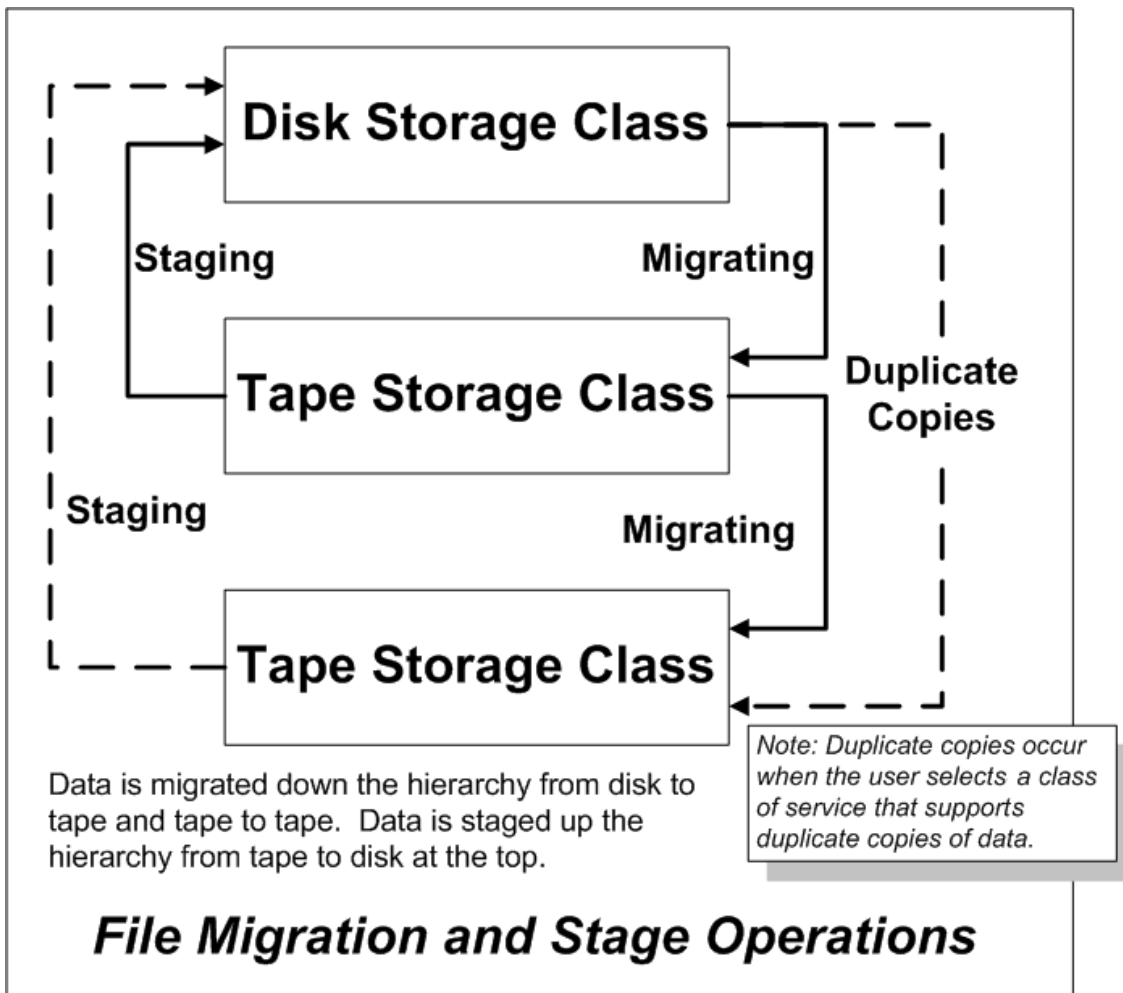


Figure 1. File migration and stage operations

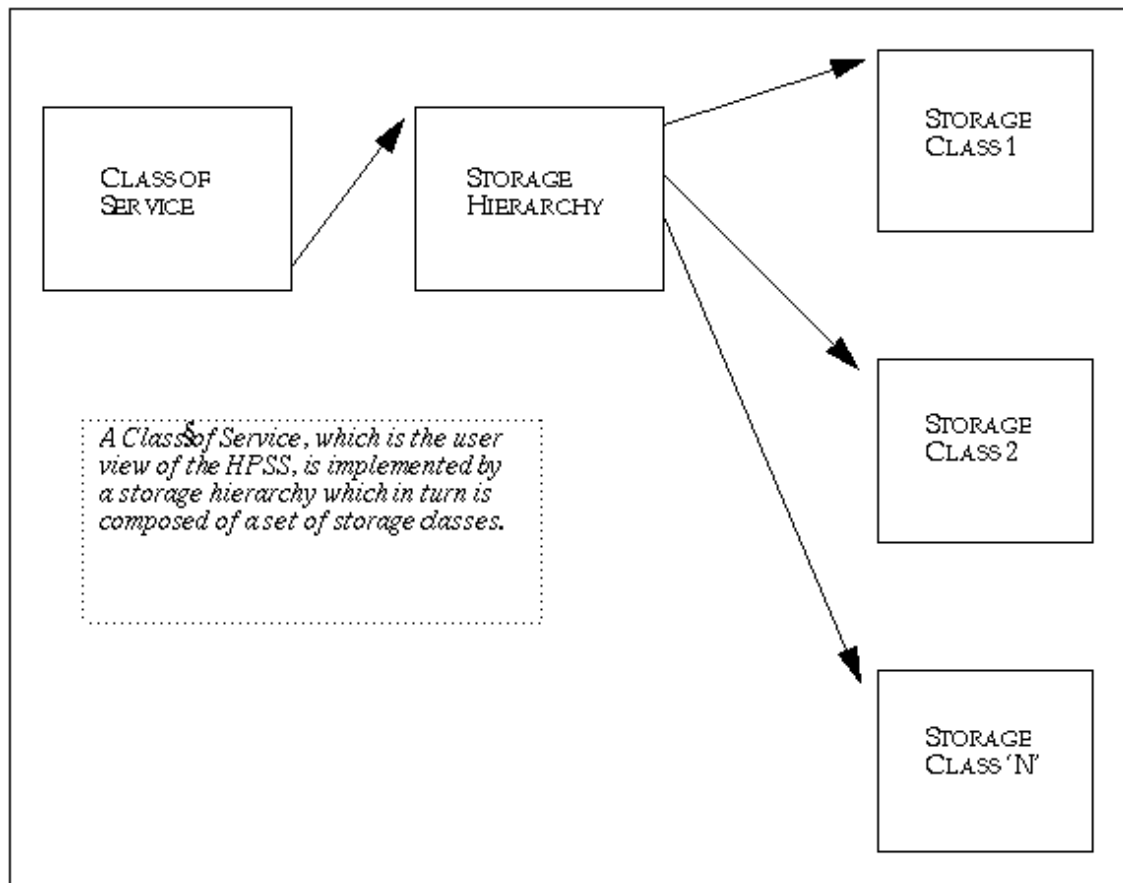


Figure 2. Class of Service/hierarchy/storage class

### 2.3.1. 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.

### 2.3.2. 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](#).

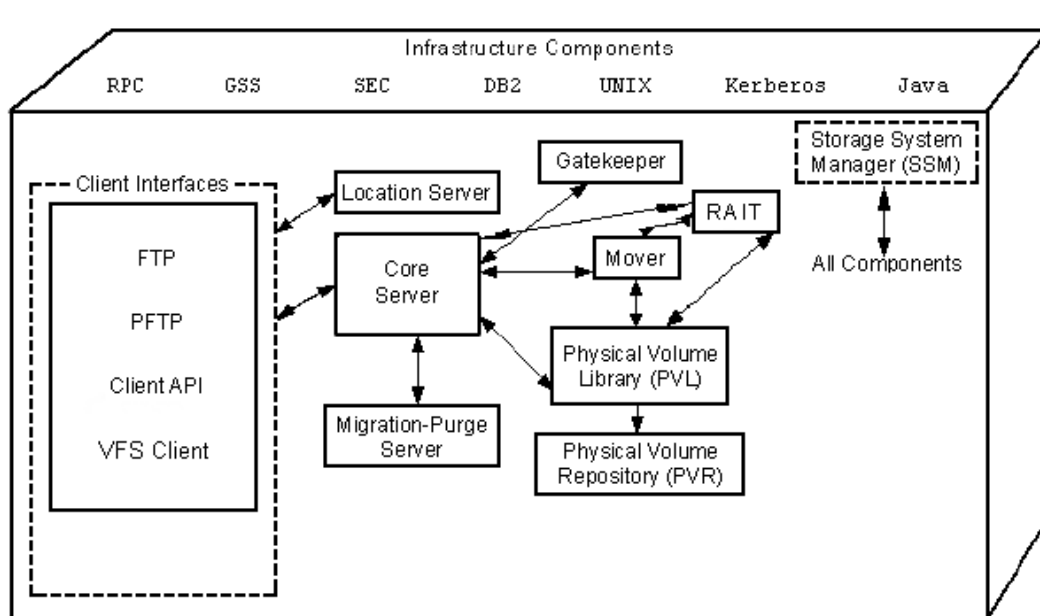


Figure 3. HPSS components

#### 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** 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 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.

### **2.3.3. 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.

### **2.3.4. 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, and Alarms and Event messages may 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**. See [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](#).

### 2.3.5. 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 the *HPSSFS-FUSE Administrator's Guide* bundled with HPSSFS-FUSE for more information.

## 2.3.6. HPSS management interfaces

HPSS provides a graphical user interface, the SSM **hpssgui**, for HPSS administration and operations GUI. The **hpssgui** simplifies the management of HPSS by organizing a broad range of technical data into a series of easy-to-read graphic displays. The **hpssgui** 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**. This tool does not provide all the functionality of the **hpssgui**, 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** provides some rudimentary mass configuration support by means of the ability to issue configuration commands from a batch script.

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.

## 2.3.7. 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 through 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.

## 2.4. HPSS hardware platforms

### 2.4.1. 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.

### 2.4.2. 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 AIX and Linux are not on the HPSS distribution image. Maintenance of the PFTP and Client API software on platforms other than AIX and 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 ( <i>see Note 1</i> )
IBM AIX		X	X	X
Oracle Solaris x86		X	X	X
RHEL (x86)	X	X	X	X
RHEL (Power PC)	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.

### **2.4.3. 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 3. HPSS planning

---

## 3.1. 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. We recommend that the administrator read this entire chapter before planning the system.*

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

### 3.1.1. 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.

# HPSS Generic Configuration

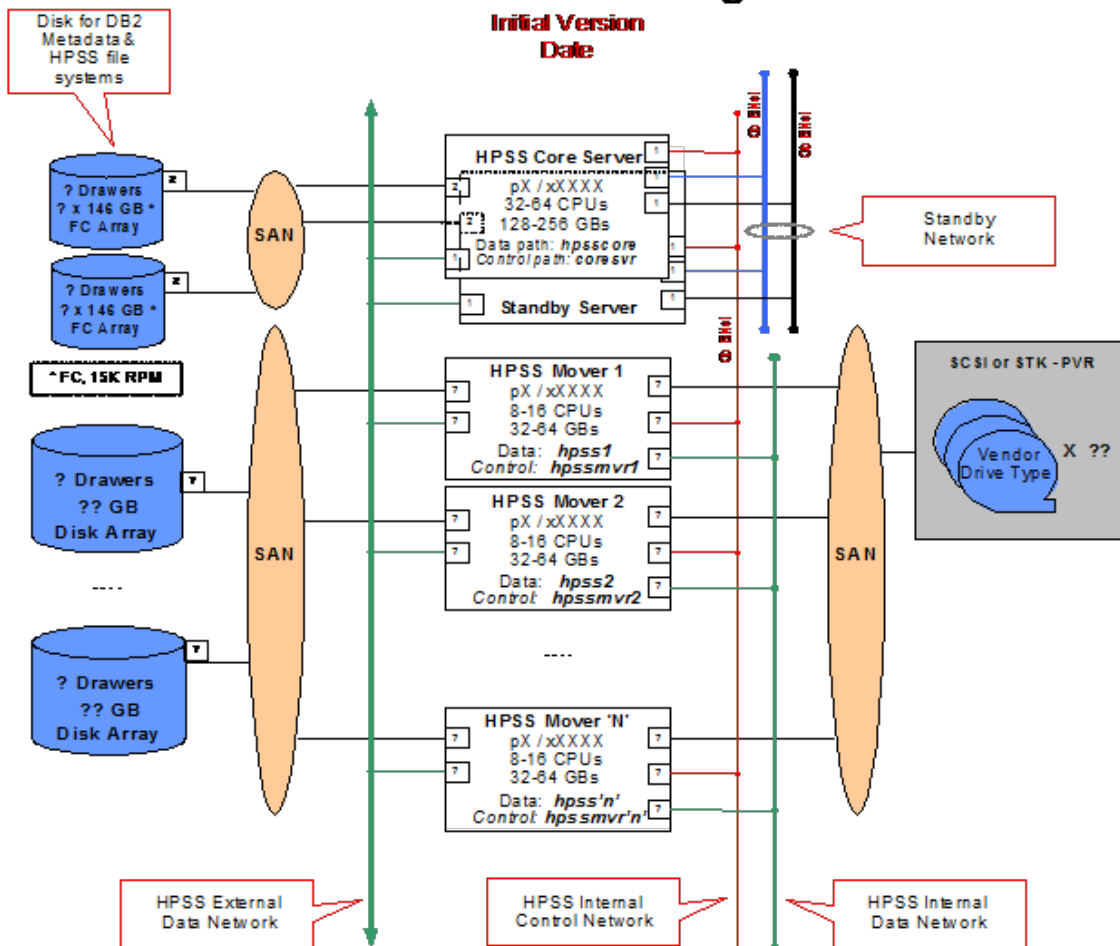


Figure 4. HPSS generic configuration

## 3.1.2. 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. 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.
  5. 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.
  6. 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.
7. Define the accounting policy to be used. Refer to [Accounting policy and validation](#) for more information on the accounting policy configuration.
  8. Define the logging policy for each of the HPSS servers. Refer to [Logging policy](#) for more information on the logging policy configuration.
  9. Define the security policy for the HPSS system. Refer to [Security policy](#) for more information on the security policy for HPSS.
  10. 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.
  11. 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.

### 3.1.3. 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 required software that will be needed to run HPSS.

### 3.1.4. 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 of the for more information on setting up the security level for an SSM user.

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.

### 3.1.5. 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.

## 3.2. 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.

### 3.2.1. 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.

### **3.2.2. 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.

### **3.2.3. 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.

### **3.2.4. 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.

### **3.2.5. 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.

### **3.2.6. 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.

### 3.2.7. 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 `hpsssm`. (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.

#### 3.2.7.1. 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** binary must be built using the distributed source code. However, it is the site's responsibility to obtain the necessary Kerberos components.

ACLs 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.

### 3.2.8. 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 admin 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.

## 3.3. 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.

### 3.3.1. 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.

### 3.3.1.1. 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>.

### 3.3.1.2. OpenSSL

HPSS uses the OpenSSL Project's cryptographic library for securing UNIX authentication. OpenSSL comes bundled with AIX and Linux. Sites using Kerberos with AIX will need an updated version of the standard AIX package of OpenSSL. Refer to <http://www.openssl.org/> for additional information about OpenSSL.

### 3.3.1.3. 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. Sites running AIX will need to install the appropriate level of Kerberos as an additional step. If building Kerberos from source on AIX, contact HPSS support if you need additional guidance.

### 3.3.1.4. 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. This option is supported only on AIX. The fileset for the IBM Kerberos client is located on the AIX Expansion Pack CD.

OpenLDAP is necessary for Kerberos authentication on Linux. OpenLDAP uses MIT Kerberos, even on AIX, 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.

### 3.3.1.5. Java

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

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

<http://www.ibm.com/developerworks/java/jdk/aix/service.html>

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

### 3.3.1.6. 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.

### 3.3.1.7. 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 RHEL 8.x, the required version is 2.11.

### 3.3.1.8. 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.

### 3.3.1.9. Apache Commons Codec

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

## 3.3.2. 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.

### 3.3.2.1. 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.*

#### **3.3.2.2. HPSS client nodes**

This section describes the prerequisite requirements for running HPSS clients.

#### **SSM client requirements**

The client node where the SSM **hpssgui** and **hpssadm** applications run must meet the following requirements:

- Supported platforms: AIX, Linux, Windows

#### **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.

## **3.4. Hardware considerations**

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

### **3.4.1. 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.

### 3.4.2. 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.

#### 3.4.2.1. 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.

### 3.4.2.2. Oracle StorageTek

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

### 3.4.2.3. 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.

## 3.4.3. 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.

## 3.4.4. 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), 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.*

### 3.4.4.1. 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*

<b>Cartridge type</b>	<b>Drive preference list</b>	<b>Operation supported</b>
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



<b>Cartridge type</b>	<b>Drive preference list</b>	<b>Operation supported</b>
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
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

Cartridge type	Drive preference list	Operation supported
STK 9840D	STK 9840D	R/W
STK 9940A	STK 9940A STK 9940B	R/W R
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

### 3.4.5. 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.

### 3.4.6. 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.

### 3.4.7. Special bid considerations

The following options are available by special bid only:

- HPSS High Availability

## 3.5. 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.

### 3.5.1. 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.

### 3.5.2. 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.

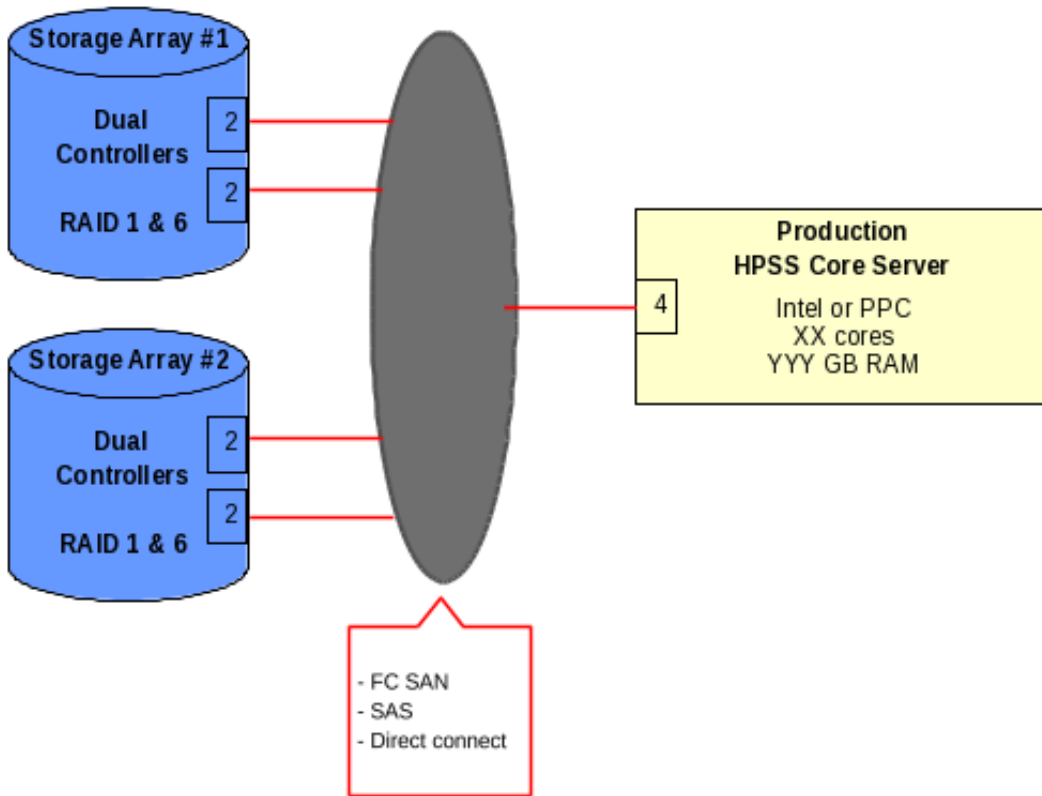


Figure 5. HPSS Core Server and metadata resources

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.

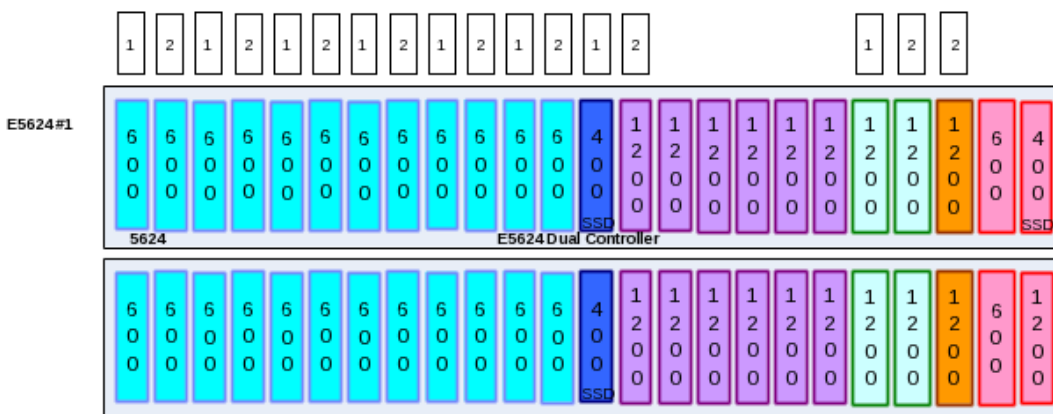


Figure 6. Metadata disk layout - Rack 1

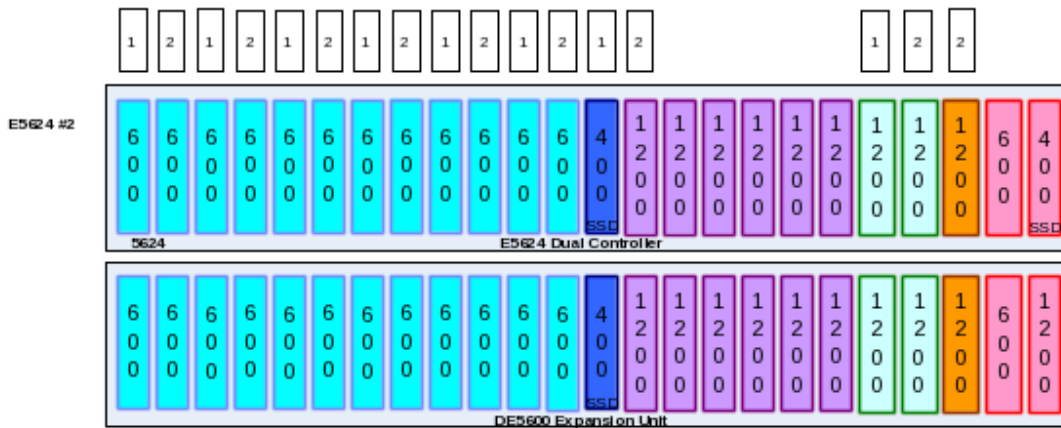


Figure 7. Metadata disk layout - Rack 2

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.

### 3.5.2.1. HPSS and DB2 file systems

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

#### **/hpss\_src**

The HPSS software is installed in the `/hpss_src` directory. A symbolic link `/opt/hpss` will link to a subdirectory under `/hpss_src` where the appropriate HPSS version has been installed.

#### **/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 `/var/hpss` 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; 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.

### **3.5.2.2. 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, **mkhps** will store the DB2-related files for HPSS in the `/db2data/db2_hpsfdb` 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.

### 3.5.2.3. 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

<b>HPSS and DB2 file systems</b> <b>There is one DB2 server</b>				
LV name or device	VG	Size *	File system mount point	Symlink to file system
varhpss	hpss_vg	64 GB	/varhpss	/var/hpss
hpss_src	hpss_vg	64 GB	/hpss_src	
optdb2	hpss_vg	64 GB	/optdb2	/opt/ibm/db2
hpss_corefile	hpss_vg	64 GB	/hpss_corefile	/var/hpss/ad m/core
db2_hpssdb	hpss_vg	64 GB	/db2data/db2_hpssdb	
db2_p0000_backup1	db2_p0000_backup1_vg	10.7 TB	/db2data/p0000/db2_ backup1	
db2_p0000_backup2	db2_p0000_backup2_vg	10.7 TB	/db2data/p0000/db2_ backup2	
db2_p0000_log	db2_p0000_log_vg	361 GB	/db2data/p0000/db2_l og	
db2_p0000_logmirror	db2_p0000_logmirror_vg	361 GB	/db2data/p0000/db2_l ogmirror	
db2_p0000_logarch1	db2_logarch1_vg	1094 GB	/db2data/p0000/db2_l ogarchive1	
db2_p0000_logarch2	db2_logarch2_vg	1094 GB	/db2data/p0000/db2_l ogarchive2	
db2_p0000_stg0001	db2_p0000_stg0001_vg	545 GB	/db2data/p0000/stg00 01	
db2_p0000_stg0002	db2_p0000_stg0002_vg	545 GB	/db2data/p0000/stg00 02	
...	...	...	...	
db2_p0000_stg0024	db2_p0000_stg0024_vg	545 GB	/db2data/p0000/stg00 24	

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_p0000_stg0001	db2_p0000_stg0001_vg	db2_p0000_stg0002	db2_p0000_stg0002_vg
db2_p0000_stg0003	db2_p0000_stg0003_vg	db2_p0000_stg0004	db2_p0000_stg0004_vg
...	...	...	...
db2_p0000_stg0023	db2_p0000_stg0023_vg	db2_p0000_stg0024	db2_p0000_stg0024_vg
db2_p0000_log	db2_log_vg	db2_p0000_logmirror	db2_p0000_logmirror_vg
db2_p0000_backup1	db2_backup1_vg	db2_p0000_backup2	db2_p0000_backup2_vg
hpssfs1-1	hpss_vg	hpssfs2-1	hpss_vg
hpssfs1-2	hpss_vg	hpssfs2-2	hpss_vg
db2_p0000_logarch1	db2_p0000_logarch1_vg	db2_p0000_logarch2	db2_p0000_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_p0000_stg0001	0	db2_p0000_stg0001	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_stg0003	1	db2_p0000_stg0003	594 GB	1+1 RAID1	DB2 Storage
...	...	...	...	...	...
db2_p0000_stg0023	11	db2_p0000_stg0023	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_log	12	db2_p0000_log	394 GB	1+1 RAID1	DB2 Log
db2_p0000_backup1	13	db2_p0000_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_p0000_logarch1	16	db2_p0000_logarch1	1194 GB	1+1 RAID1	DB2 Log Archive

Table 7. Storage Array #2

Array	Vol ID	Volume label	~Size	Config	Purpose
db2_p0000_stg0002	0	db2_p0000_stg0002	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_stg0004	1	db2_p0000_stg0004	594 GB	1+1 RAID1	DB2 Storage
...	...	...	...	...	...
db2_p0000_stg0024	11	db2_p0000_stg0024	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_log	12	db2_p0000_log	394 GB	1+1 RAID1	DB2 Log

Array	Vol ID	Volume label	~Size	Config	Purpose
db2_p0000_backup2	13	db2_p0000_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_p0000_logarch2	16	db2_p0000_logarch2	1117 GB	1+1 RAID1	DB2 Log Archive

### 3.5.3. System memory and disk space

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

#### 3.5.3.1. 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.

#### 3.5.3.2. System disk space requirements for running SSM

The SSM Graphical User Interface, **hpssgui**, and Command Line Interface, **hpssadm**, have an option to create session log files. **hpssgui** records all status bar and pop-up messages issued during the session in its log. **hpssadm** 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** and **hpssadm** startup scripts. See their man pages for details.

#### 3.5.3.3. 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*

<b>Amount of physical memory</b>	<b>Minimum recommended amount of paging space</b>
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

## 3.6. HPSS interface considerations

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

### 3.6.1. 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*.

### 3.6.2. 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.

### 3.6.3. 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**, 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.

## 3.7. 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.

### 3.7.1. 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*

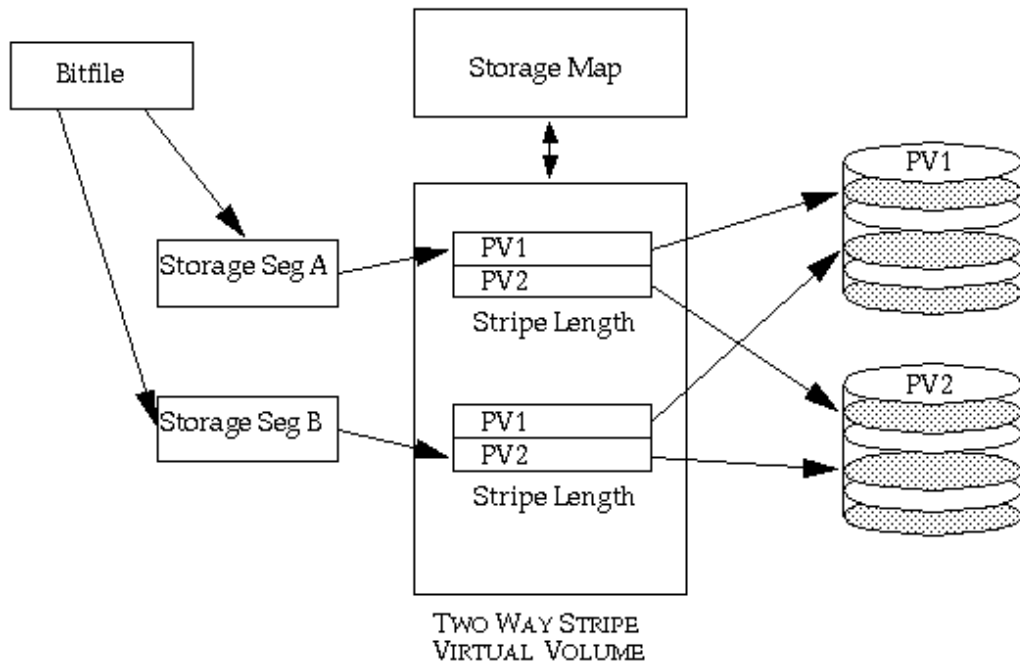
The HPSS Core Server maintains the HPSS name space in system metadata. Refer to [HPSS sizing considerations](#) for details on sizing the name space. Refer to the [DB2 space shortage](#) section for information on handling a metadata space shortage. By utilizing multiple storage subsystems, it is possible to distribute large name spaces across multiple Core Servers. Junctions between the name 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](#).



RELATIONSHIP BETWEEN BITFILES, SEGMENTS, MAPS, VV AND PV

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.

#### 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.

### **3.7.2. 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** 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 parallelizes the migration of files from disk to tape. The number of bytes that the MPS migrates simultaneously is user configurable via 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 to four different virtual volumes at one time.

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.

### 3.7.3. 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 known 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.

### **3.7.4. 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.

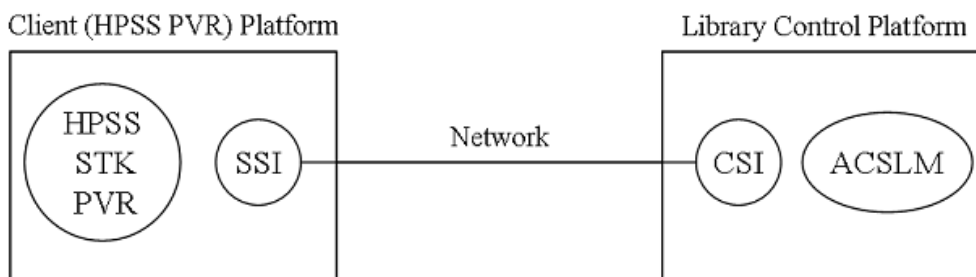
### 3.7.5. PVR

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.

#### 3.7.5.1. 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.

ACSLM 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>`.



### 3.7.5.2. 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.

### 3.7.5.3. SCSI PVR

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

## 3.7.6. 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.

### 3.7.6.1. 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.

### 3.7.6.2. 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
```

### 3.7.6.3. 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.

### 3.7.7. 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.

### 3.7.8. 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.

### 3.7.9. Storage System Management

SSM has three components:

- A System Manager (**hpss\_ssmsm**), which communicates with all other HPSS components requiring monitoring or control.

- The GUI (**hpssgui**), 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**), 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.

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

Starting up the SSM GUI (**hpssgui**) 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**) directly on the server UNIX platform, and this is the recommended configuration.

Both the SSM GUI client, **hpssgui**, and the SSM Command Line client, **hpssadm**, 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.

Variable	Default value	Functionality
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.
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.

### 3.8. 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.

## 3.9. Storage policy considerations

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

### 3.9.1. 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.

#### 3.9.1.1. 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.

#### 3.9.1.2. 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.

### 3.9.2. 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** utility and released when the **recover** 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** 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.

### 3.9.3. 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**) manual page 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.

### 3.9.4. 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.

#### **3.9.4.1. 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.

#### **3.9.4.2. 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.

#### **3.9.4.3. 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.

#### **3.9.4.4. 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.

### **3.9.5. 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.

### 3.9.6. 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
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.

## 3.10. 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](#).

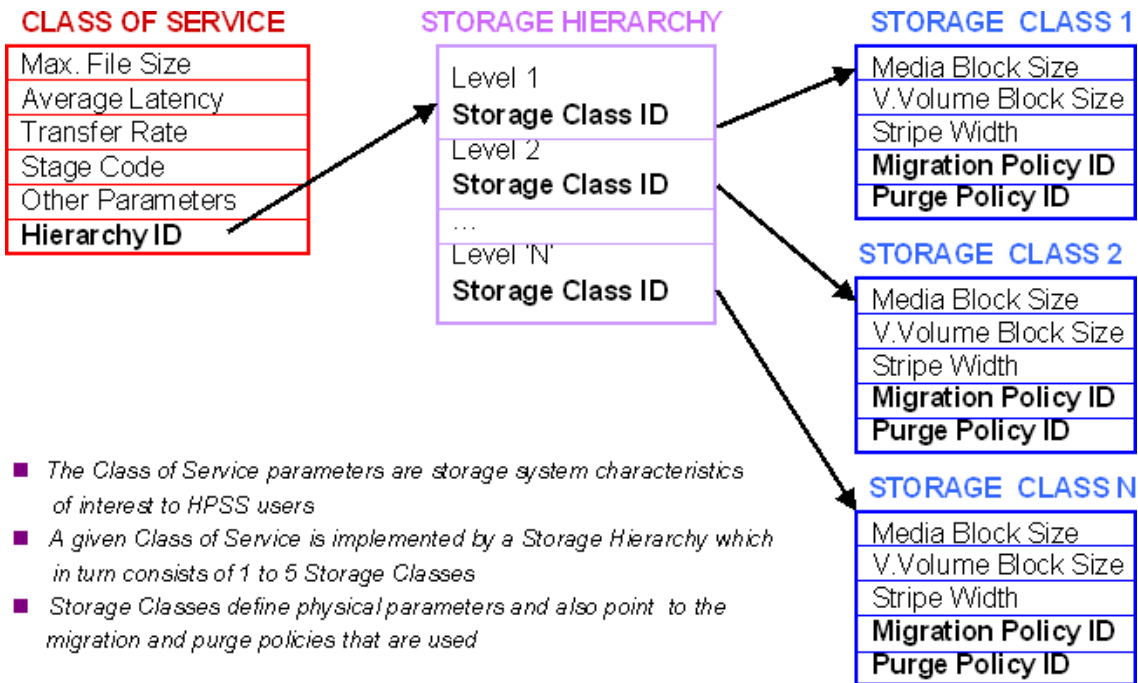


Figure 9. Relationship of Class of Service, storage hierarchy, and storage class

### 3.10.1. 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.

### 3.10.1.1. 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.

### 3.10.1.2. 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.

### 3.10.1.3. 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.

### 3.10.1.4. 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** and **recover** 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:

$$(<\text{File Size in MB}>/10) + 20$$

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:

$$(<\text{File Size in MB}>/20) + (2 \times 20)$$

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.

#### **3.10.1.5. 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](#).

#### **3.10.1.6. 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.

### 3.10.1.7. 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.

### 3.10.1.8. 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.

### 3.10.1.9. 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.

### 3.10.1.10. 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.

### 3.10.1.11. 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.

### 3.10.1.12. 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*

<b>Disk type</b>	<b>Media block size</b>	<b>Minimum access size</b>	<b>Minimum virtual volume block size</b>
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

<b>Tape type</b>	<b>Media block size</b>	<b>Blocks between tape marks</b>	<b>Estimated physical volume size</b>
IBM 3580 (LTO)	256 KB	2700	100 GB
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 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)

<b>Tape type</b>	<b>Media block size</b>	<b>Blocks between tape marks</b>	<b>Estimated physical volume size</b>
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)
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.

### 3.10.2. 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.

### 3.10.3. 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.

### 3.10.3.1. 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**.

### 3.10.3.2. 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.

### 3.10.3.3. 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.

#### 3.10.3.4. 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.

#### 3.10.3.5. 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.

### 3.10.3.6. 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.

### 3.10.3.7. 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.

## 3.10.4. 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.

## 3.11. HPSS performance considerations

This section provides some additional hints for enhancing HPSS performance.

### 3.11.1. 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 V10.5 InfoCenter at : <http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp>.

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

### 3.11.2. 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.

### 3.11.3. 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.

### 3.11.4. 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.

### 3.11.5. 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](#).

### 3.11.6. 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.

### 3.11.7. 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.

### 3.11.8. 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.

### 3.11.9. 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.

### 3.11.10. HPSSFS-FUSE interface

Refer to the *HPSSFS-FUSE Administrator's Guide*, bundled with the HPSSFS-FUSE RPM.

## 3.12. 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.

### **3.13. 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.



# Chapter 4. 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.*

## 4.1. 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, 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.
- Configure the SSH prerequisite software on the core HPSS server node (at a minimum) and configure SSH to accept connections from IBM Houston. Include the Houston subnet IP addresses 192.94.47 and 12.39.169 in the local firewall routing rules, as necessary.
- 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.

## 4.2. Set up file systems

The following sections describe how to set up the various file systems used by HPSS and DB2.

### 4.2.1. 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`.

### 4.2.2. 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.

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.

## 4.3. Set up tape libraries

### 4.3.1. 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.

### 4.3.2. 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* 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.

## 4.4. 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.

### 4.4.1. 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):

```
% ioccheck -w -t 20 -b 1mb /dev/nst1
```

To measure the maximum-compressed write performance on st1 (and then rewind the tape):

```
% ioccheck -w -t 20 -f 0 -b 1mb /dev/st1
```

To measure read performance on drive st1 using the previously-written uncompressed and compressed files:

```
% ioccheck -r -t 20 -b 1mb /dev/nst1  
% ioccheck -r -t 20 -b 1mb /dev/nst1
```

To empty the tape:

```
% mt -f /dev/st1 rewind  
% mt -f /dev/st1 weof 2
```

## 4.5. Set up disk drives

### 4.5.1. 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.

## 4.6. 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: nis dns files
```

- For each AIX Ethernet network interface, verify that the **en0** and **et0** interfaces are not both configured at the same time (we recommend only using **en0** unless the other nodes in the network are all using the **802.3 et\*** interface). Configure the local name service with the unique hostname for each network interface on all nodes and verify that each hostname is resolvable from other nodes.
- 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*

<b>Network option</b>	<b>Description</b>
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. The recommendation from AIX development is to set this value to at least two times the maximum number of concurrent connections times the size of the socket send/receive buffers. The default setting for AIX 4.3.2 and later is the smaller of (1) half the size of physical memory or (2) 1 GB.
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 UDP/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.

### 4.6.1. HPSS.conf configuration file

The `HPSS.conf` configuration file contains tuning options to be used by HPSS clients and servers.

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.

## 4.7. 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.

## 4.8. 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 =  $67,108,864 \times 1,024 = 68,719,476,736$

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\\_](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
```

## 4.9. 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 5. 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](#)

[Prepare post-installation procedures](#)

[Locate HPSS documentation and set up manual pages](#)

[Define HPSS environment variables](#)

[Tune DB2](#)

## 5.1. Prepare for installation

The following sections discuss the steps that need to be taken in order to prepare the system for HPSS installation.

### 5.1.1. Distribution media

Obtain the HPSS software from HPSS support.

### 5.1.2. 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.

### 5.1.3. 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 [Prepare post-installation procedures](#) for more information on the HPSS file ownerships and permissions.*

### 5.1.4. Installation target directory preparation

By default, the HPSS software is installed in the `/hpss_src` directory. Before installing the HPSS software, make sure that the installation target directory satisfies the following conditions:

- The `/hpss_src` 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.*

## 5.2. 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](https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=docs:release_notes).

### 5.2.1. Install Java

Java is required to compile HPSS software. It is also required to run the SSM Command Line Interface (**hpssadm**). 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** application to run directly on the client machine to maximize **hpssgui** 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.

### 5.2.2. 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*.

### 5.2.3. 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*.

### 5.2.4. 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.

### 5.2.5. 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.

### 5.2.6. 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`.

### 5.2.7. 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.

```
% 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/V10.5/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. i

## 5.3. 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. <http://hpss-collaboration.org>

Installation of HPSS with RPMs is simple. RPM installation is done with:

```
rpm -ivh <rpm-name>
```

There are eleven main RPM packages for HPSS:

Table 15. RPM packages

Package	Description
hpss-lib	This is a dependency for all the other RPMs except for the hpss-doc, hpss-src, and hpss-stk-src RPMs. 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.ksh
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-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. <i>This RPM is only available at special request.</i>
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.
hpss-src	This contains all the HPSS source files with the exception of files in test directories and IBM confidential STK files. <i>This RPM is only available at special request.</i>
hpss-stk-src	This contains the IBM confidential STK source files. <i>This RPM is only available at special request.</i>

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).

## 5.4. Install HPSS

This section details the steps necessary to install HPSS on the root, Movers, and other client nodes.

### 5.4.1. On core

To install HPSS on a Core Server machine, the RPMs `hpss-lib` and `hpss-core` should be installed and a link set up for `/opt/hpss`, 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_src/hpss-X.Y-0
hpss-core-X.Y-0.el8
ln -sf /opt/ibm/db2/default /opt/hpss_src/hpss-X.Y-0/db2
Files for package hpss-core installed under
/opt/hpss_src/hpss-X.Y-0

# unlink /opt/hpss
# ln -s /opt/hpss_src/hpss-X.Y-0 /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.

### 5.4.2. On Mover

To install HPSS on a Mover machine, the RPMs `hpss-lib` and `hpss-mvr` should be installed and a link set up for `/opt/hpss`, 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
/hpss_src/hpss-X.Y-0
hpss-mvr-X.Y-0.el8
Files for package hpss-mvr installed under
/hpss_src/hpss-X.Y-0

# unlink /opt/hpss
# ln -s /hpss_src/hpss-X.Y-0 /opt/hpss
```

### 5.4.3. On client

To install HPSS on a client machine, the RPMs `hpss-lib` and `hpss-clnt` should be installed and a link set up for `/opt/hpss`, 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
/hpss_src/hpss-X.Y-0
hpss-clnt-X.Y-0.el8
Files for package hpss-clnt installed under
/hpss_src/hpss-X.Y-0

# unlink /opt/hpss
# ln -s /hpss_src/hpss-X.Y-0 /opt/hpss
```

### 5.4.4. On remote PVR

To install HPSS on a remote PVR machine, the RPMs `hpss-lib` and `hpss-pvr` should be installed and a link set up for `/opt/hpss`, 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
/hpss_src/hpss-X.Y-0
hpss-pvr-X.Y-0.el8
ln -sf /opt/ibm/db2/default /hpss_src/hpss-X.Y-0/db2
Files for package hpss-pvr installed under
/hpss_src/hpss-X.Y-0

# unlink /opt/hpss
# ln -s /hpss_src/hpss-X.Y-0 /opt/hpss
```

### 5.4.5. 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
```

### 5.4.6. 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.

## 5.5. 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.

### 5.5.1. 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.

### 5.5.2. 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

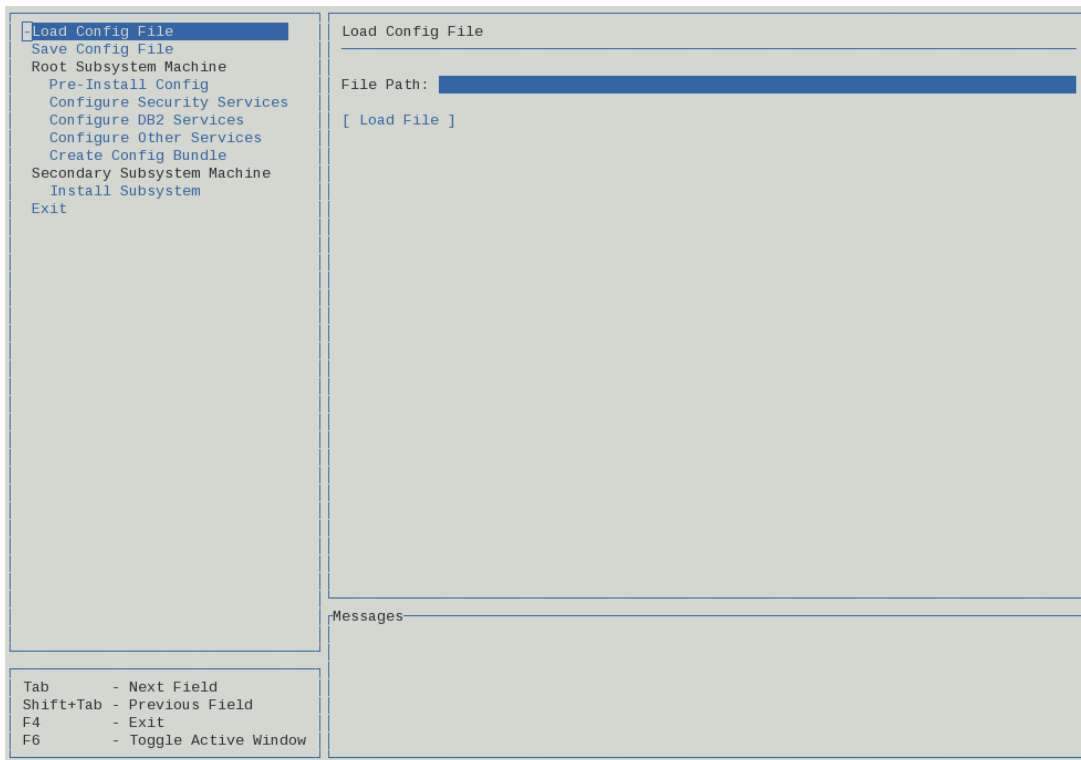
### Create configuration bundle

The following sections describe the procedures to perform these steps.

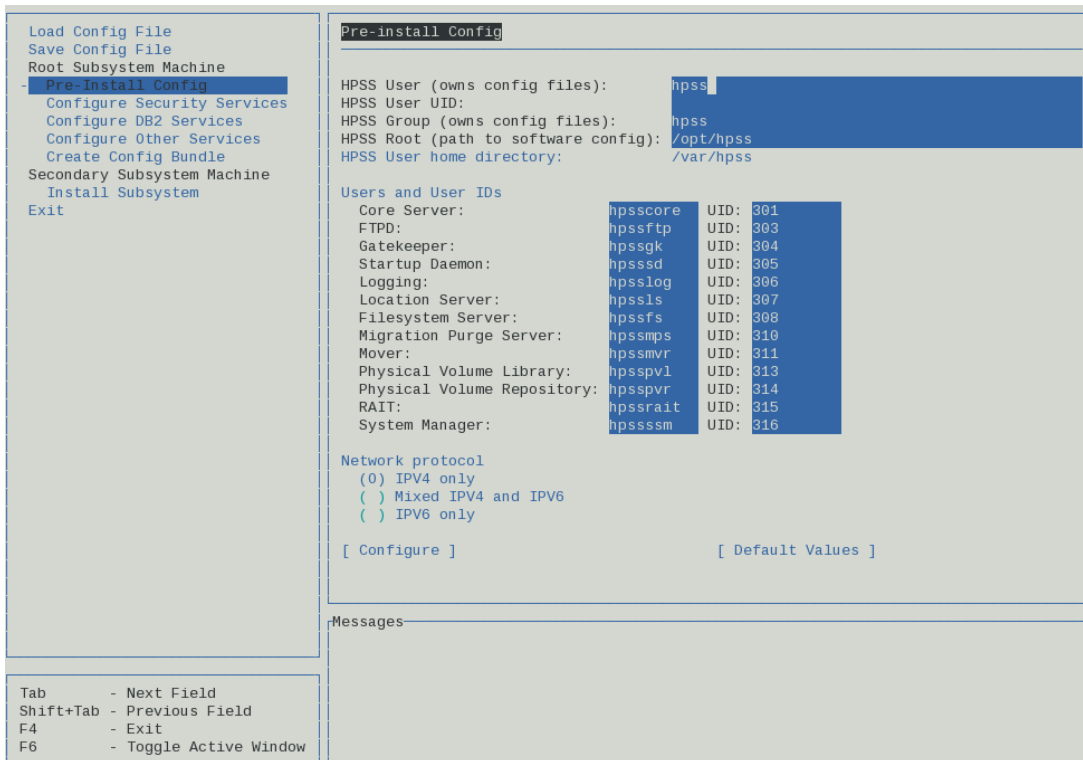
#### 5.5.2.1. Pre-installation configuration

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, this may be **/hps\_src/hps-11.1-0/**, in which case the **mkhpss** tool should be in **/hps\_src/hps-11.1-0/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:



- 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`.
- 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.

### 5.5.2.2. Configure HPSS security services

This section describes the authentication and authorization mechanisms which can be configured to enable HPSS to provide the desired security services. The following authentication and authorization combinations are supported:

Table 16. Supported authentication plus authorization methods

Authentication mechanism	Authorization mechanism
UNIX	UNIX
Kerberos	UNIX
Kerberos	LDAP

The following sections describe the procedure to configure the above security combinations using



the **mkhpss** utility.

### Configure UNIX authentication and UNIX authorization

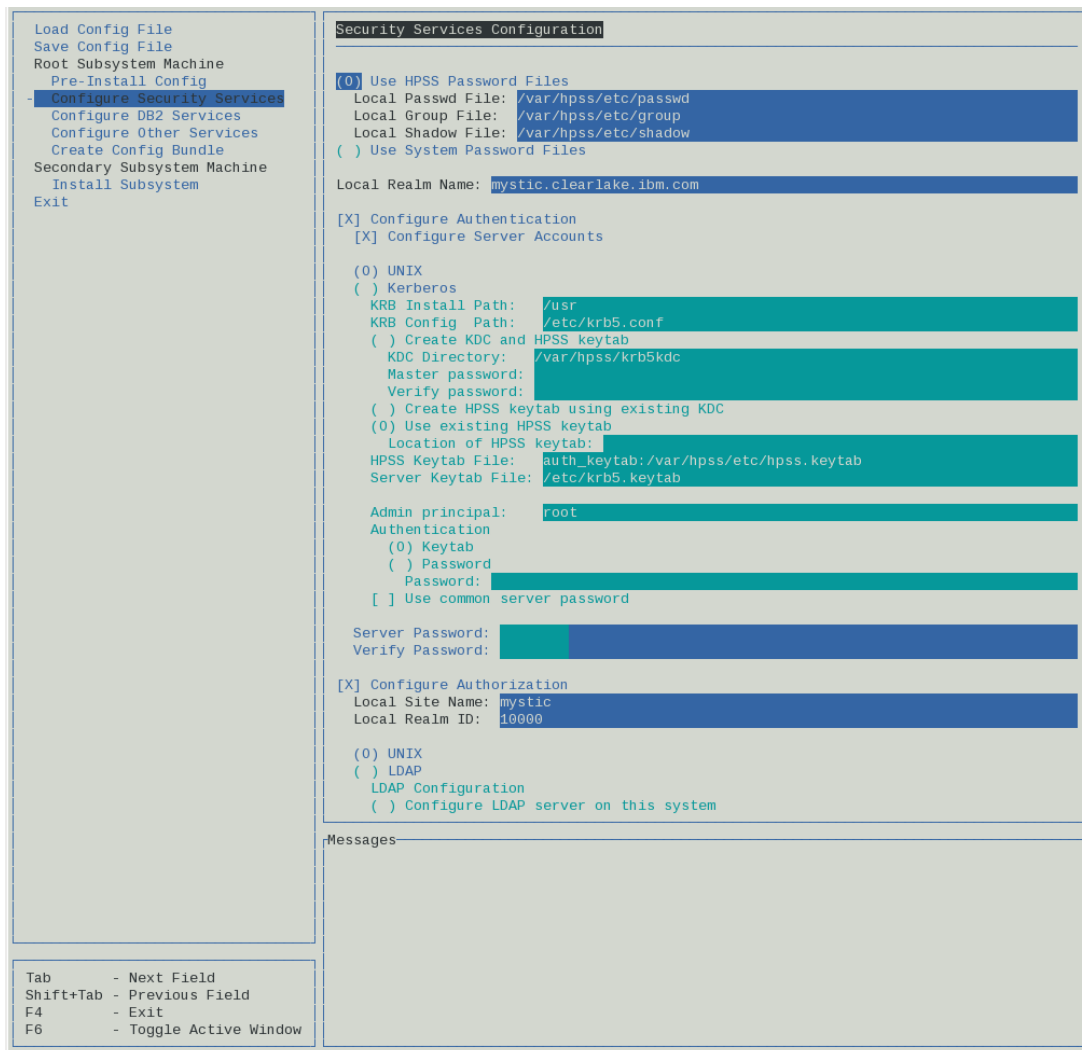
From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Security Services" option. Perform the following steps using the content panel:

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`, and `/var/hpss/etc/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`).

#### Local Realm Name

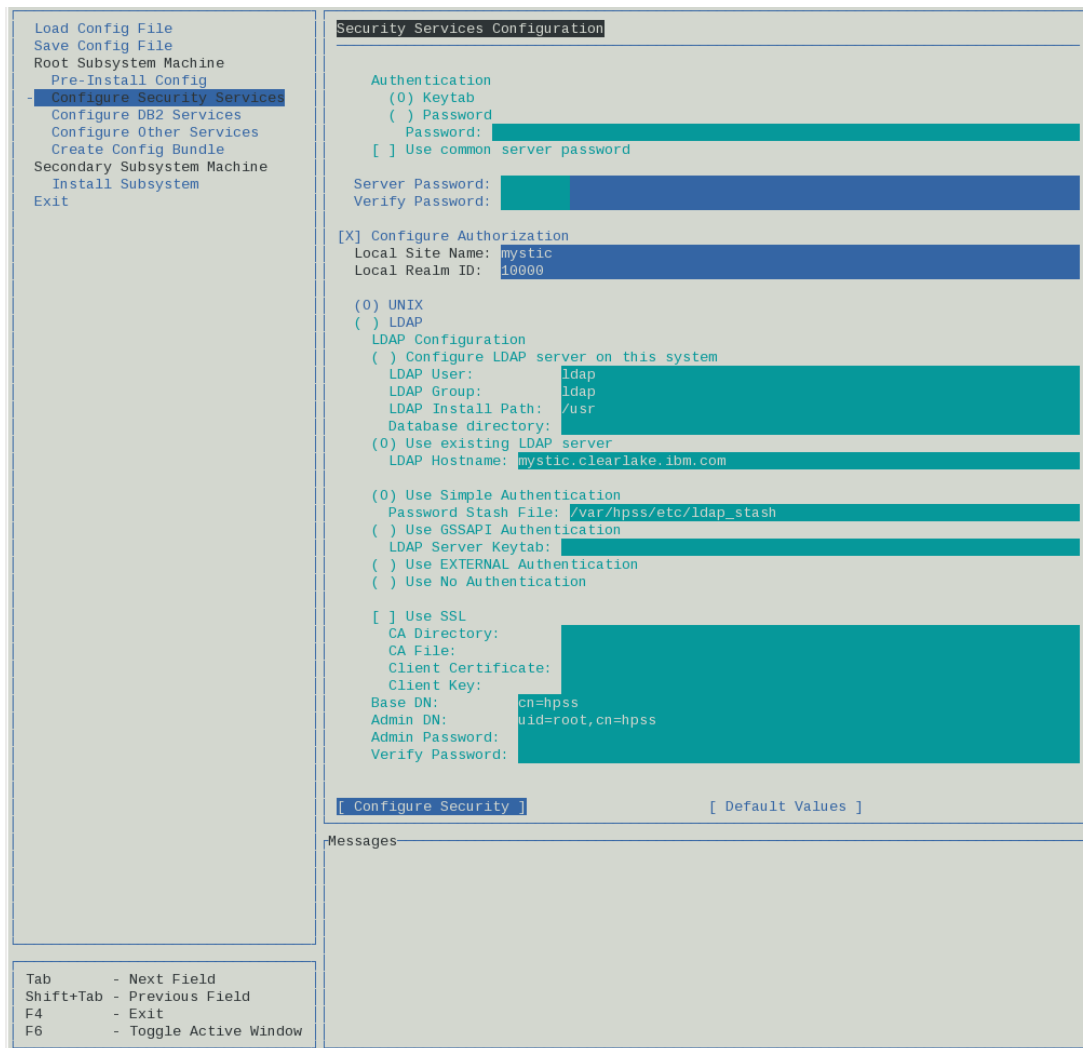
By convention, this value is usually set to the fully qualified domain name of the local host.

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:

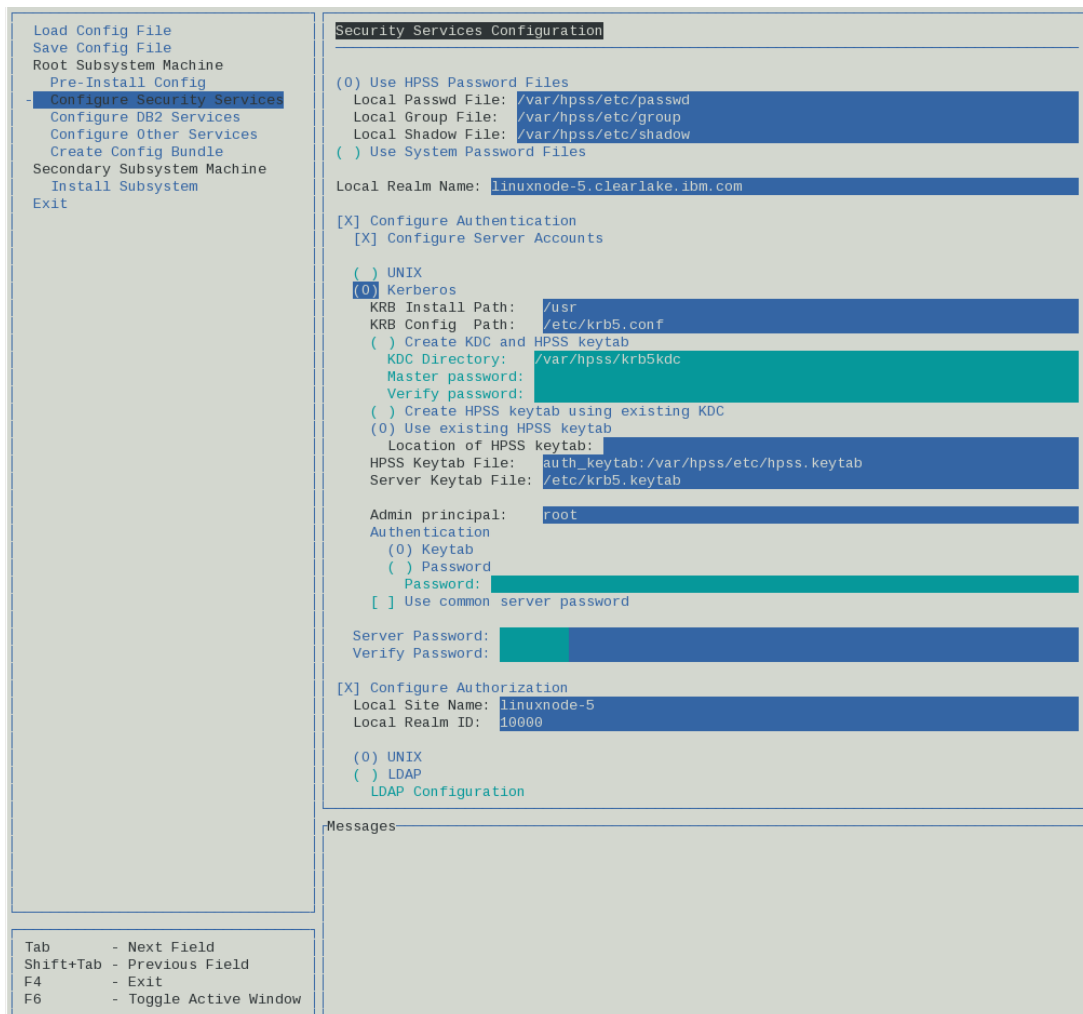
```
run command finished: mkhpss_run_core_sec_config
```

This will be followed by a timestamp and the duration of the command.

### 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`, and `/var/hpss/etc/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`). 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 **mkhps** 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, **mkhps** 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/hps/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.

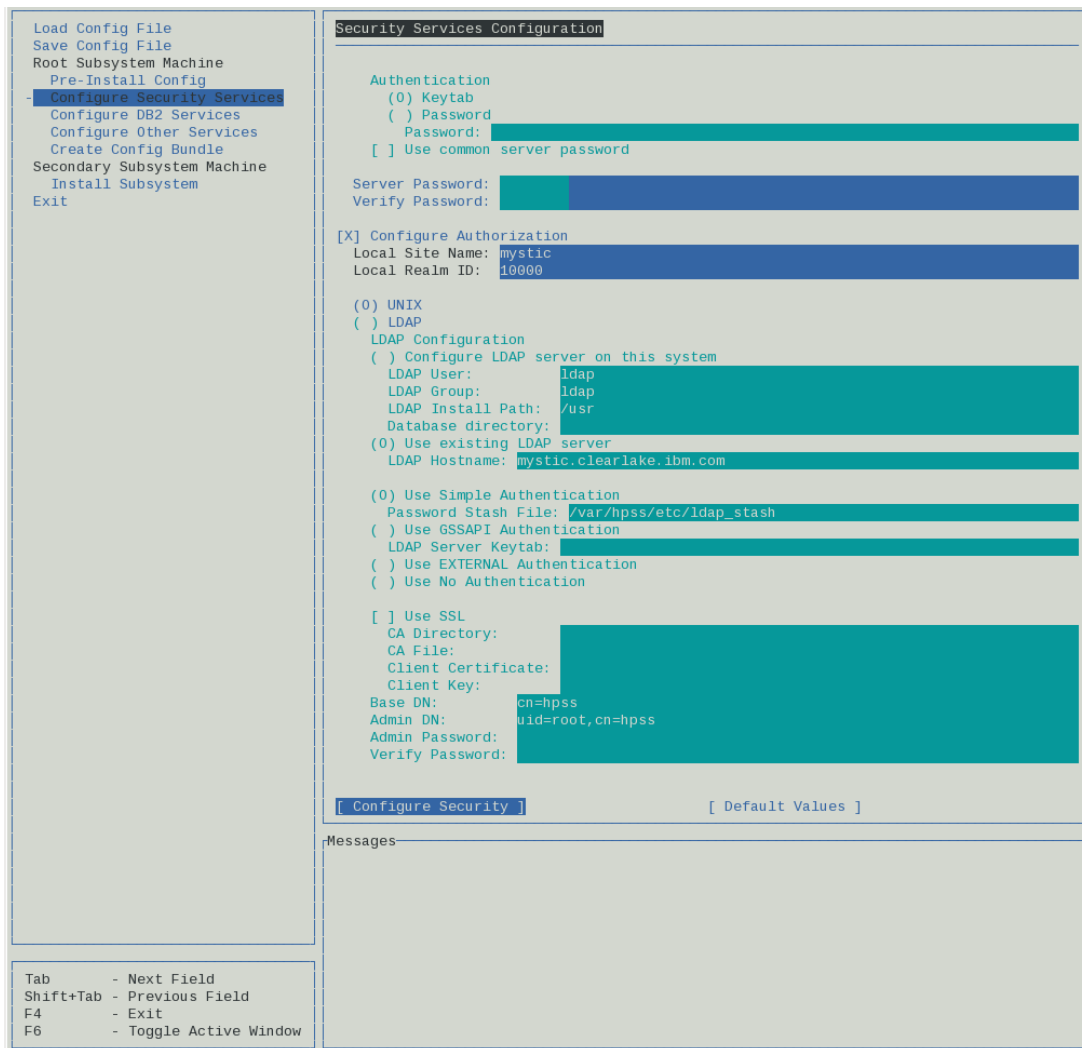
### Keytab File

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:

Load Config File  
Save Config File  
Root Subsystem Machine  
Pre-Install Config  
**Configure Security Services**  
Configure DB2 Services  
Configure Other Services  
Create Config Bundle  
Secondary Subsystem Machine  
Install Subsystem  
Exit

**Security Services Configuration**

Use HPSS Password Files  
Local Passwd File: /var/hpss/etc/passwd  
Local Group File: /var/hpss/etc/group  
Local Shadow File: /var/hpss/etc/shadow  
 Use System Password Files

Local Realm Name: linuxnode-5.clearlake.ibm.com

Configure Authentication  
 Configure Server Accounts

UNIX  
 Kerberos  
KRB Install Path: /usr  
KRB Config Path: /etc/krb5.conf  
 Create KDC and HPSS keytab  
KDC Directory: /var/hpss/krb5kdc  
Master password: [redacted]  
Verify password: [redacted]  
 Create HPSS keytab using existing KDC  
 Use existing HPSS keytab  
Location of HPSS keytab: [redacted]  
HPSS Keytab File: auth\_keytab:/var/hpss/etc/hpss.keytab  
Server Keytab File: /etc/krb5.keytab

Admin principal: root

Authentication  
 Keytab  
 Password  
Password: [redacted]  
 Use common server password

Server Password: [redacted]  
Verify Password: [redacted]

Configure Authorization  
Local Site Name: linuxnode-5  
Local Realm ID: 10000

UNIX  
 LDAP  
LDAP Configuration

Messages

Tab - Next Field  
Shift+Tab - Previous Field  
F4 - Exit  
F6 - Toggle Active Window

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 `mkhps` 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 **mkhps** 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, **mkhps** 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/hps/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.

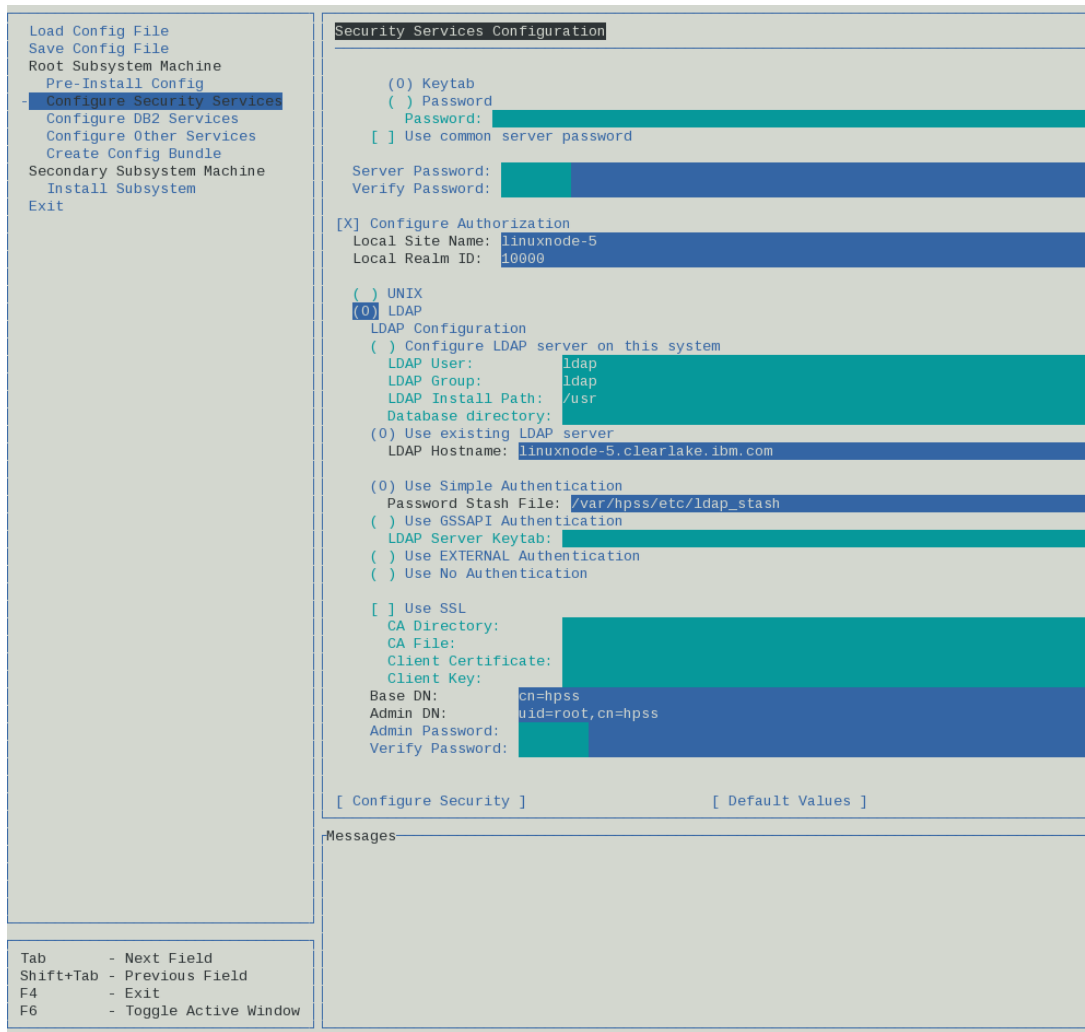
**Keytab File**

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.

**5.5.2.3. 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:

```
DB2 Configuration

DB2 Install Path: /opt/ibm/db2/default
Instance Owner: hpssdb
Schema Name: HPSS
Number of Partitions: 1

DB2 Authorization Group
for HPSS Servers: hpsssrvr
User Name for Servers: hpss

[X] Create DB2 Instance
Instance Owner Password:
Verify Password:
Instance Owner Group: hpssdb
Owner Home Directory: /db2data/db2_hpssdb
Instance client port number: 59999
Starting Partition Port Number: 60000

(0) Use SSH for database admin
( ) Use RSH for database admin

[X] Setup DB2 Authentication
Password:
Verify Password:

[X] Create Config Database
Config Database Alias: cfg
Primary Log Directory: /db2data/db2_log/cfg
[X] Mirrored Log
Directory: /db2data/db2_logmirror/cfg
[X] Archive Log 1
Directory: /db2data/db2_logarchive1/cfg
[X] Archive Log 2
Directory: /db2data/db2_logarchive2/cfg
```

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/p0000/db2_log/cfg`. 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/p0000/db2_logmirror/cfg`.

## 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/p0000/db2_logarchive1/cfg`.

## 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 `/db2/data/p0000/db2_logarchive2/cfg`.

3. Continuation of DB2 Configuration fields are as displayed on the screen shown:

```
( ) Custom DDL
DDL File:
Warning: Keep Custom DDL unchecked and DDL File path empty unless directed by
HPSS Support. Accidentally selecting Custom DDL without providing a valid
DDL file results in tables not being created.

Database path: /db2data/db2_hpssdb
Tablespace paths: /db2data/p $4N /stg0001,/db2data/p $4N /stg0002
Warning: Tablespace paths input will scroll with long inputs
All tablespace paths must contain at least one partition expression
"$4N" delimited by a space on both sides. No space before.
or after the comma.
```

## 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. Each tablespace path used must contain at least one partition expression "\$4N" delimited by a space on both sides. No spaces before or after the comma if more than one path is used.

Eg `/db2data/p $4N /stg0001,/db2data/p $4N /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.

+ NOTE: **Tablespace Paths** use [DB2 Partition Expressions](#). This should generally use the  $\$4N$  format to spread partitions across available file system resources.  $\$4N$  will put a directory, per partition, under the path specified. For example, `/db2data/p $4N /stg0001` would create a directory per partition. If the *Number of Partitions* was set to "1" with one storage path file system, then DB2 would expect `/db2data/p0000/stg0001` to exist. If the *Number of Partitions* was set to "4" and the storage file system is 4, then the same expression will result with four directories that look like this: `/db2data/p0000/stg0001`, `/db2data/p0001/stg0002`, `/db2data/p0002/stg0003`, `/db2data/p0003/stg0004`.

### Create Subsys Database

By default the check box to create "subsys" databases is selected.

```
[X] Create Subsys Databases
Subsys Database Alias Base: subsys
Number of Subsystems: 1
Primary Log Directory: /db2data/p $4N /db2_log/ $S
[X] Mirrored Log
  Directory: /db2data/p $4N /db2_logmirror/ $S
[X] Archive Log 1
  Directory: /db2data/p $4N /db2_logarchive1/ $S
[X] Archive Log 2
  Directory: /db2data/p $4N /db2_logarchive2/ $S
Warning: $S means 'substitute the subsystem name' and does not refer to a partition

Extent size: 32
(0) Automatic Storage
( ) Database-Managed File Containers
  DMS File Size (MBs): 10240
( ) Custom DDL
  DDL File:
Warning: Keep Custom DDL unchecked and DDL File path empty unless directed by
HPSS Support. Accidentally selecting Custom DDL without providing a valid
DDL file results in tables not being created.

Database path: /db2data/db2_hpssdb
Tablespace paths: /db2data/p $4N /stg0001,/db2data/p $4N /stg0002
Warning: Tablespace paths input will scroll with long inputs
All tablespace paths must contain at least one partition expression
"$4N" delimited by a space on both sides. No space before.
or after the comma.

[ Configure DB2 ] [ Default Values ]
```



*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/p0000/db2_log/ $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/p0000/db2_logmirror/ $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/p0000/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/p0000/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.



**Tablespace Paths** use [DB2 Partition Expressions](#). This should generally use the  $\$4N$  format to spread partitions across available file system resources.  $\$4N$  will put a directory, per partition, under the path specified. For example, `/db2data/p $4N /stg0001` would create a directory per partition. If the *Number of Partitions* was set to "1", then DB2 would expect `/db2data/p0000/stg0001` to exist. If the *Number of Partitions* was set to "4", then the same expression will result with four directories that look like this: `/db2data/p0000/stg0001`, `/db2data/p0001/stg0002`, `/db2data/p0002/stg0003`, `/db2data/p0003/stg0004`.



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.

#### 5.5.2.4. 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 **mklhpss** 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.

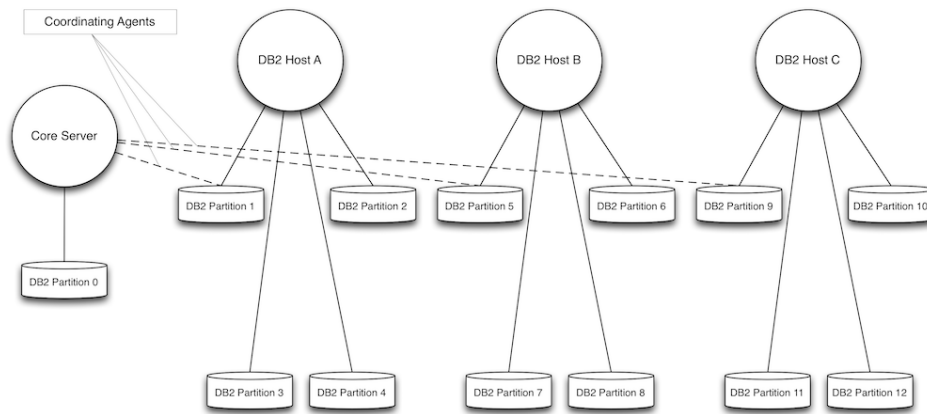


Figure 10. DB2 Off-Node Example

```
> cat /db2data/db2_hpssdb/sql1lib/db2nodes.cfg
0 hpscore 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 (note

that we used paths stg[0001-0004] in the example, but any integer n could be chosen for the number of paths per partition):

```
hpsscore:
For DB2 data:
/db2data/p0000/dbpath/hcfg
/db2data/p0000/dbpath/hsubsys1
/db2data/p0000/stg[0001-0004]
/db2data/p0000/stg[0001-0004]
For DB2 primary logs:
/db2data/p0000/db2_log/hcfg
/db2data/p0000/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p0000/db2_logmirror/hcfg
/db2data/p0000/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p0000/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p0000/db2_logarchive1
For DB2 database backup images:
/db2data/p0000/db2_backup1
/db2data/p0000/db2_backup2

db2hosta:
For DB2 data:
/db2data/p[0001-0004]/dbpath/hcfg
/db2data/p[0001-0004]/dbpath/hsubsys1
/db2data/p[0001-0004]/stg[0001-0004]

For DB2 primary logs:
/db2data/p[0001-0004]/db2_log/hcfg
/db2data/p[0001-0004]/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p[0001-0004]/db2_logmirror/hcfg
/db2data/p[0001-0004]/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p[0001-0004]/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p[0001-0004]/db2_logarchive2
For DB2 database backup images:
/db2data/p[0001-0004]/db2_backup1
/db2data/p[0001-0004]/db2_backup2

db2hostb:
For DB2 data:
/db2data/p[0005-0008]/dbpath/hcfg
/db2data/p[0005-0008]/dbpath/hsubsys1
/db2data/p[0005-0008]/stg[0001-0004]
For DB2 primary logs:
/db2data/p[0005-0008]/db2_log/hcfg
```

```

/db2data/p[0005-0008]/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p[0005-0008]/db2_logmirror/hcfg
/db2data/p[0005-0008]/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p[0005-0008]/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p[0005-0008]/db2_logarchive2
For DB2 database backup images:
/db2data/p[0005-0008]/db2_backup1
/db2data/p[0005-0008]/db2_backup2

```

```

db2hostc:
For DB2 data:
/db2data/p[0009-0012]/dbpath/hcfg
/db2data/p[0009-0012]/dbpath/hsubsys1
/db2data/p[0009-0012]/stg[0001-0004]
For DB2 primary logs:
/db2data/p[0009-0012]/db2_log/hcfg
/db2data/p[0009-0012]/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p[0009-0012]/db2_logmirror/hcfg
/db2data/p[0009-0012]/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p[0009-0012]/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p[0009-0012]/db2_logarchive2
For DB2 database backup images:
/db2data/p[0009-0012]/db2_backup1
/db2data/p[0009-0012]/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/p \ $4N /stg0001',          \
      '/db2data/p \ $4N /stg0002',          \
      '/db2data/p \ $4N /stg0003',          \
      '/db2data/p \ $4N /stg0004'          \
      dbpath on '/db2data/p \ $4N /dbpath/hcfg' \
      pagesize 8192"
DB20000I  The CREATE DATABASE command completed successfully.

> db2 "create database hsubsys1 automatic storage yes on \
      '/db2data/p \ $4N /stg0001',          \
      '/db2data/p \ $4N /stg0002',          \
      '/db2data/p \ $4N /stg0003',          \
      '/db2data/p \ $4N /stg0004'          \
      dbpath on '/db2data/p \ $4N /dbpath/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/p[0000-0012]/db2_log/hcfg
db2 update db config for hcfg DBPARTITIONNUM [0-12] using MIRRORLOGPATH
/db2data/p[0000-0012]/db2_logmirror/hcfg
db2 update db config for hcfg DBPARTITIONNUM [0-12] using LOGARCHMETH1
DISK:/db2data/p[0000-0012]/db2_logarchive1
db2 update db config for hcfg DBPARTITIONNUM [0-12] using LOGARCHMETH2
DISK:/db2data/p[0000-0012]/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/p[0000-0012]/db2_log/hsubsys1
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using MIRRORLOGPATH
/db2data/p[0000-0012]/db2_logmirror/hsubsys1
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using LOGARCHMETH1
DISK:/db2data/p[0000-0012]/db2_logarchive1
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using LOGARCHMETH2
DISK:/db2data/p[0000-0012]/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:
```

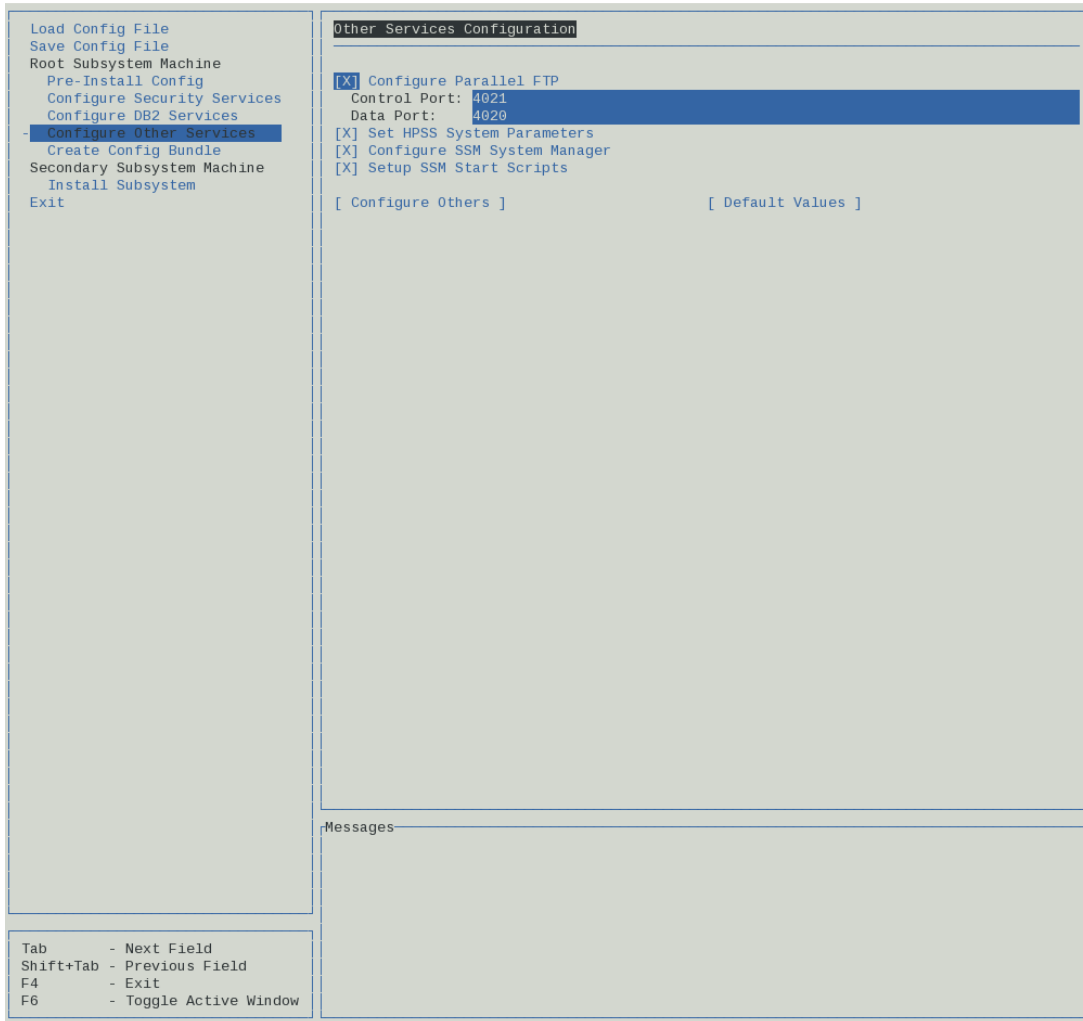
+

```
export DB2_USE_LOCAL_RESYNC=true
```

#### **5.5.2.5. 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.

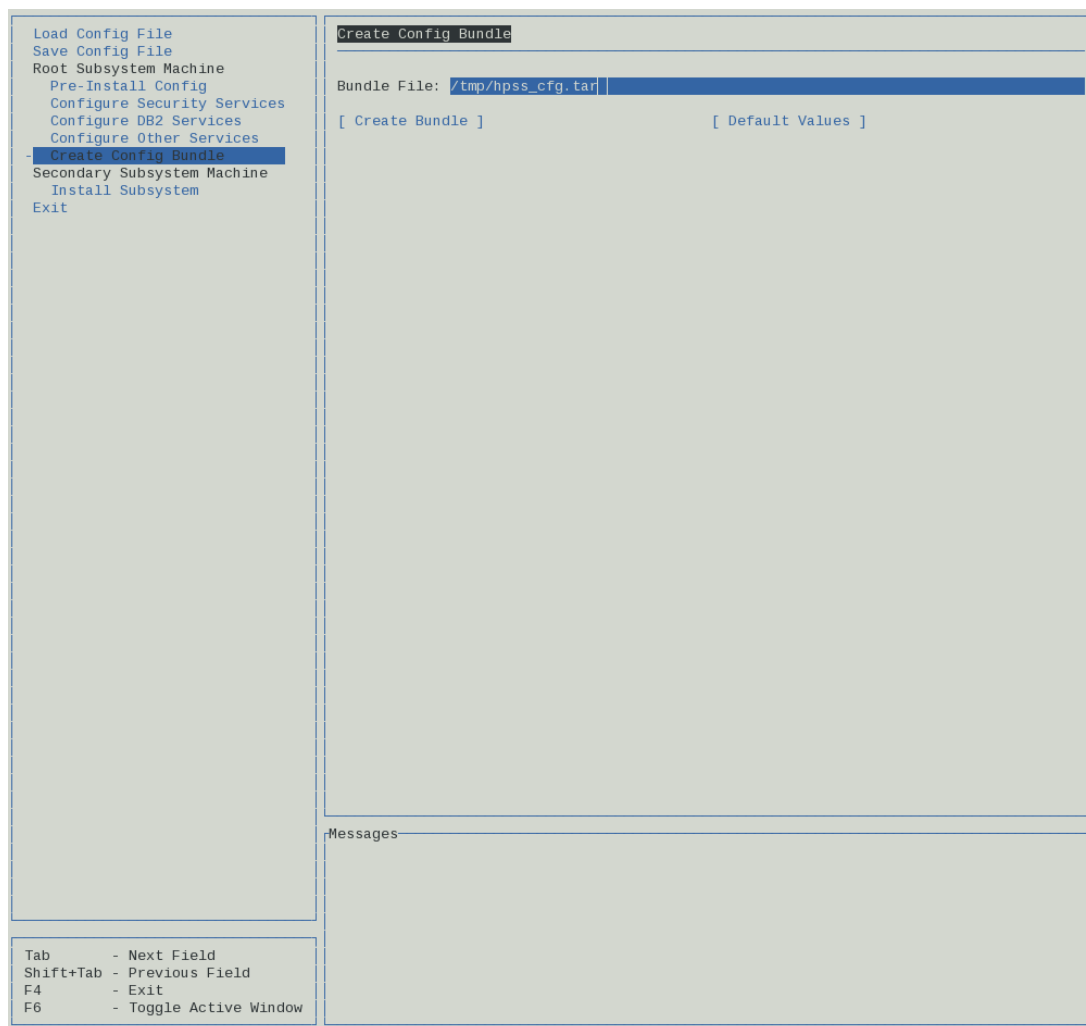
```
/var/hpss/etc/hpss.keytab  
/var/hpss/etc/mm.keytab  
/var/hpss/krb5kdc/kadm5.keytab
```

### 5.5.2.6. 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.

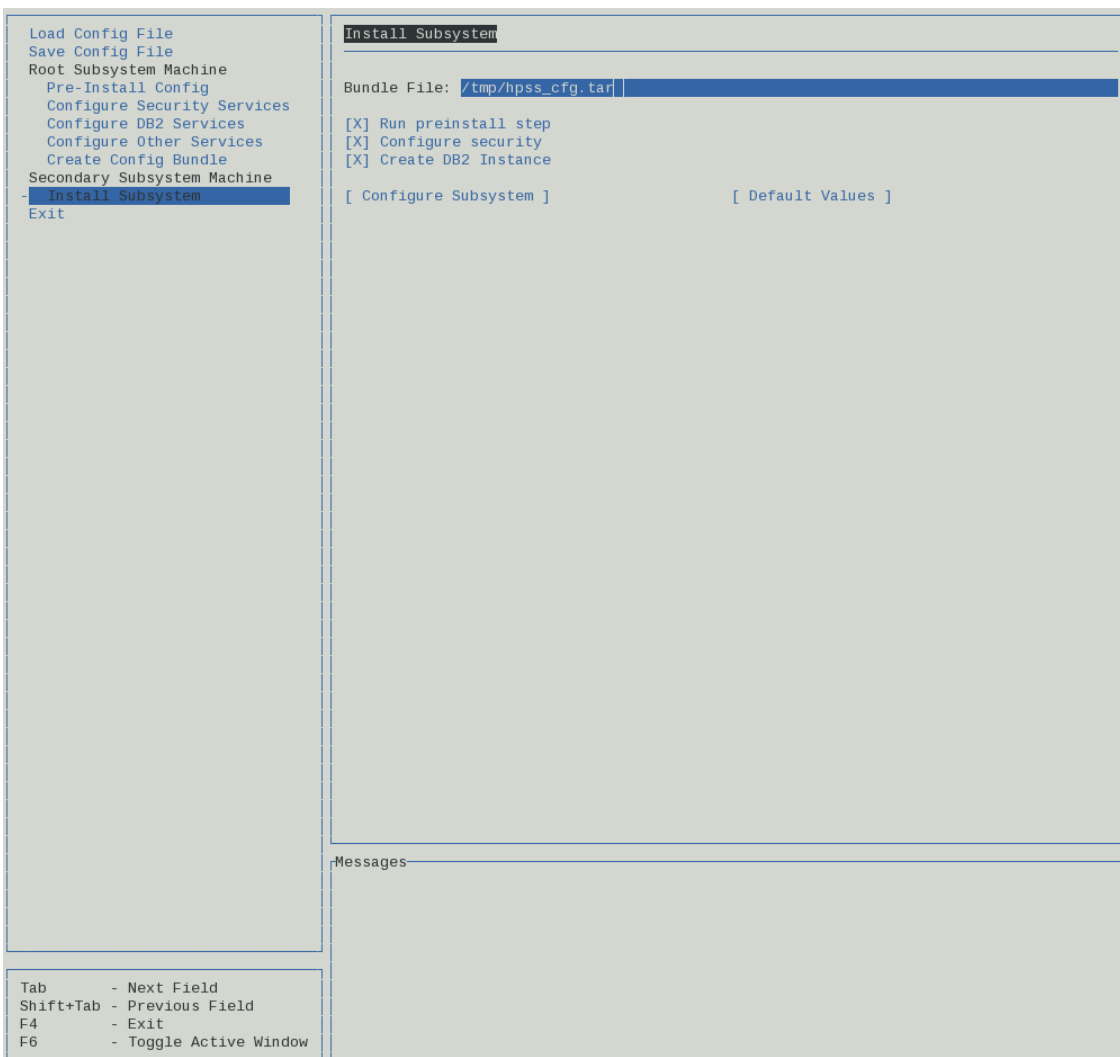
3. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_create_bundle
```

This will be followed by a timestamp and the duration of the command.

### 5.5.3. 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:



### 5.5.4. 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/V10.5.)

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/V10.5/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.

## 5.6. Prepare 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/msg/`<HPSS message catalog>

`/opt/hpss/tools/`<HPSS tools>

`/opt/hpss/man/`<HPSS man pages>

`/opt/hpss/config/`<HPSS configuration scripts>

`/opt/hpss/src/`<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	<code>rwxr-xr-x bin bin</code>
------------------	--------------------------------

Include files	<code>r--r--r-- bin bin</code> (write permission on generated files)
Library files	<code>r--r--r-- bin bin</code>
Source files	<code>r--r----- hpss hpss</code>
Make files	<code>rw-rw-r-- bin bin</code>
Configuration files	<code>rw-rw-r-x hpss hpss</code>



*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/V10.5/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.

## 5.7. 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.

### 5.7.1. 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** program is executed in order to be available in the SSM Help feature. The **hpssuser** program can be used to bundle the HTML files for delivery to the **hpssgui** host machine. The recommended installation location for the HTML files on each **hpssgui** host is `/hpss_src/hpss-<version>/ssmhelp` for Linux

platforms and `c:\hpss\doc` for Windows platforms.

## 5.7.2. 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 linked to `/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
```

## 5.8. 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
```

## 5.9. 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 hpssdb
# 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
```

#### 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 ]
```

#### 6. 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.

#### 7. 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.

## 5.10. 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 under the DB2 V10.5 InfoCenter at <http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp> 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.

## 5.11. Supporting both UNIX and Kerberos authentication for SSM

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 the 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.

## 5.12. 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  $3.4 \times 10^{38}$  IPv6 addresses. As a comparison, there are roughly  $9 \times 10^{21}$  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 17. 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.

### 5.12.1. 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 6. Installation and configuration of the Elastic (ELK) Stack

## 6.1. 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.

### 6.1.1. 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
```

## 6.1.2. 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.yml** 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>

## 6.1.3. 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.

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. To enable security, read: <https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-stack-security.html>

## 6.1.4. 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>

## 6.1.5. 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>

# 6.2. Installing the HPSS data capture components

## 6.2.1. 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



The HPSS data capture scripts are included with the HPSS Core Server RPM.

## 6.2.2. How to configure the data capture scripts

The data capture scripts should be configured as cron jobs. As the HPSS user, run **crontab -e** and

add the following entries.

```
*/5 * * * * /opt/hpss/bin/hpss_dump_server_metrics -r
*/5 * * * * /opt/hpss/bin/hpss_dump_acct -r
*/5 * * * * /opt/hpss/bin/hpssadm_hs_info -r
*/5 * * * * /opt/hpss/bin/hpssadm_sc_list -r
*/5 * * * * /opt/hpss/bin/hpssadm_servers_list -r
*/5 * * * * /opt/hpss/bin/hpssadm_devices_and_drives_list -r
```

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.



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.

## 6.3. Setup the HPSS dashboards

### 6.3.1. 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.

### 6.3.2. 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.

## 6.4. 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** 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\_stats**

This index contains the output from the **hpss\_server\_metrics -t stats** 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 7. 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](#).

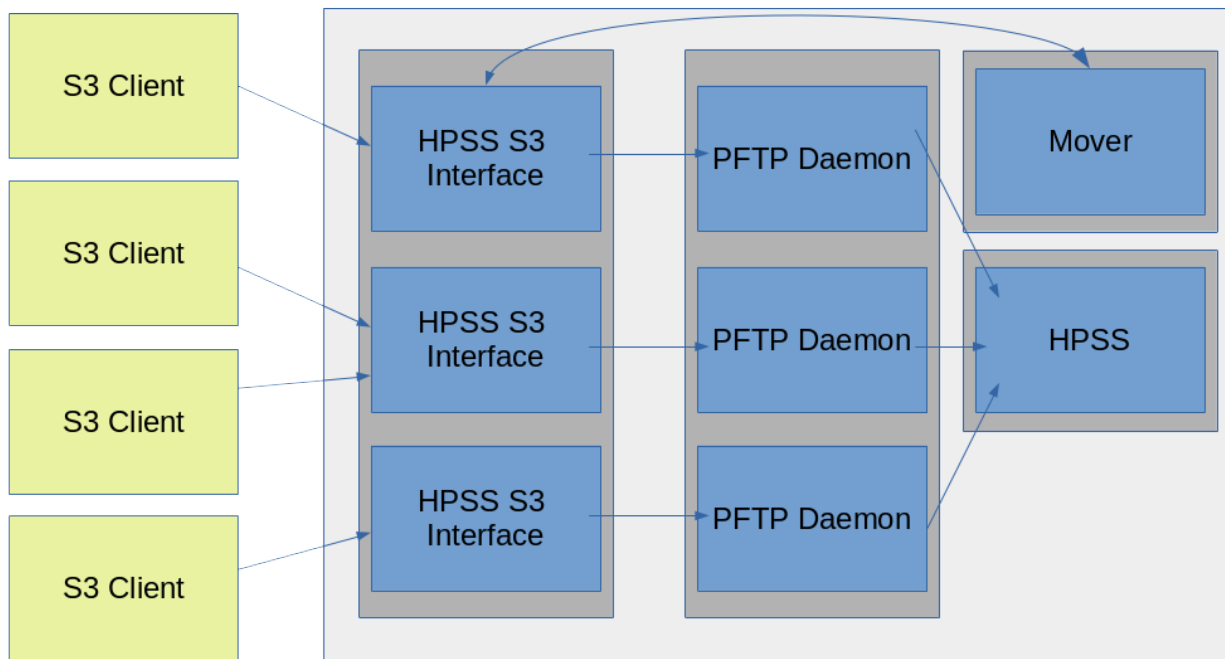
## 7.1. Overview

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.

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: hpsshps Re-enter new password: hpsshps
Enter the password to be stored: hpsspassword Re-enter password: hpsspassword
```

You will add the following properties to `hpss_s3proxy.conf` `s3proxy.hpss-creds = /var/hpss/etc/hpsss3keystore` `s3proxy.hpss-ks-password = hpsshps`

## 7.2. 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.

<b>Configuration Item</b>	<b>Description</b>	<b>Default</b>
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.hpss-creds	Credential keystore	/var/hpss/etc/hpsss3keystore
s3proxy.hpss-ks-password	Keystore password	hpsshps
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	
jclouds.pftp.hpss.cos	Default HPSS COS	<Default COS>
jclouds.pftp.basedir	Base Directory	<HPSS Mount>
jclouds.pftp.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.recv.size	HPSS PFTP I/O Buffer Size	524288
--------------------------	---------------------------	--------

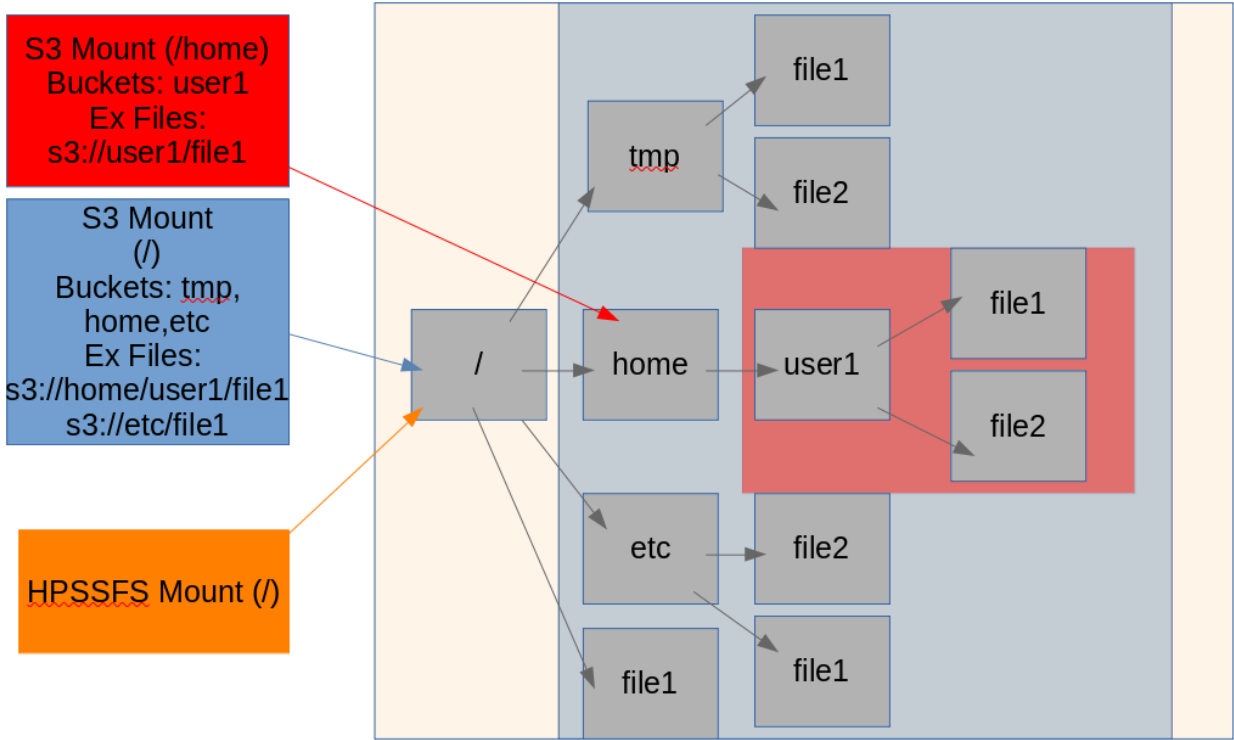
Set up the service file in systemd and start the service. Note that multiple services can be configured for different servers with different configurations, with minor tweaks to the provided service file.

```
# 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.

### 7.3. 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.

## 7.4. 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.

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.

+ 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.

## 7.5. 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.

## 7.6. 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.virtual-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.

## 7.7. 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.pftp.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).

## 7.8. 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.

### 7.8.1. 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.

### 7.8.2. 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](#).

### 7.8.3. 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)
```

## 7.9. 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.

## 7.10. 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 indicating 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
'
```

## 7.11. 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.

## 7.12. 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.

## 7.13. 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:

<code>--negotiate</code>	Uses the GSS/Negotiate mechanism, which becomes Kerberos
<code>--delegation always</code>	Tells GSS to delegate your credentials to the server
<code>-u :</code>	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 8. HPSS configuration overview

## 8.1. 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.

## 8.2. 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. 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** 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** 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](#).

## 8.3. 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](#))

## 8.4. 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 **scrub** 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** 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
```

## 8.5. 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).

## 8.6. Verification checklists (all sites)

This section provides a number of checklists regarding configuration, operational, and

performance issues.

### 8.6.1. 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** 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](#))

## **8.6.2. 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.

### **8.6.3. 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 9. Security and system access

---

## 9.1. 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.

### 9.1.1. 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. See the man page for `site.conf` for details.

## 9.1.2. Security mechanisms

HPSS supports UNIX and Kerberos mechanisms for authentication. It supports LDAP and UNIX mechanisms for authorization.

### 9.1.2.1. 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 Logon 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 Logon 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` is used to define "keytab" files that can be used to acquire credentials recognized by the UNIX authentication mechanism.
- `hpss_unix_user` is used to manage users in the HPSS password file (`/var/hpss/etc/passwd`).
- `hpss_unix_group` is used to manage users in the HPSS groups file (`/var/hpss/etc/group`).
- `hpss_unix_passwd` is used to change passwords of users in the HPSS password file.
- `hpss_unix_auth` 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*

- `hpss_unix_service` is the login service program which is launched by `rc.hpss`

### 9.1.2.2. 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**) 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.

#### 9.1.2.3. Kerberos 5



*The capability to use MIT Kerberos authentication is provided in HPSS 7.4; however, IBM Service Agreements for HPSS do not provide support for problem isolation nor fixing defects (Level 2 and Level 3 support) in MIT Kerberos. Kerberos maintenance and support must be site-provided.*

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.

#### 9.1.2.4. LDAP



*LDAP authorization is not supported by IBM Service Agreements. The following information is provided for sites planning to use LDAP authorization with HPSS 7.4 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 IBM Tivoli Directory Server (Kerberos plug-in available for AIX only) and OpenLDAP (Kerberos plug-in available for AIX and Linux). 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** 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** 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.

## 9.2. 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** utility. If, for some reason, an ACL should need to be modified in some other way, the **hpss\_server\_acl** utility can be used. See the **hpss\_server\_acl** man page for more information.

## 9.3. 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**. Security levels may be viewed and modified with the **hpss\_server\_acl** utility. See also [SSM user authorization](#).

## 9.4. Restricting user access to HPSS

## 9.4.1. 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.

## 9.4.2. 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.

### 9.4.2.1. 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.

### 9.4.2.2. 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.

### 9.4.3. 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.

### 9.4.4. 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.

### 9.4.5. Restriction Types

Individual restrictions are listed for each user in the restrict access config file. The restriction names and meanings are:

<b>RestrictAll</b>	restrict the user from all actions
<b>RestrictFileCopies</b>	restrict the user from copying files
<b>RestrictFileCreates</b>	restrict the user from creating new files
<b>RestrictFileWrites</b>	restrict the user from writing to new and existing files
<b>RestrictStages</b>	restrict the user from staging files from lower levels (typically tape) to higher levels (typically disk) in storage hierarchies

### 9.4.6. 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 10. Using SSM

---

## 10.1. The SSM System Manager

### 10.1.1. 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` utility. See the `hpss_set_env` man page for more information.

To start the SM, invoke the `rc.hpss` script as follows:

```
% su -  
% /opt/hpss/bin/rc.hpss -m start
```

### 10.1.2. 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 (`hpssgui` and `hpssadm`). 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 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 Interface Controls tab of the System Manager's *Core Server Configuration* window (see [Interface controls](#)).

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 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.



### 10.1.3. 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 **hpssgui** and **hpssadm** clients must access to reach the System Manager.

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 either `/usr/bin` (AIX) or `/usr/sbin` (Linux).

## 10.2. Quick startup of hpssgui

We recommend that **hpssgui** sessions be invoked from the user's desktop computer instead of on the HPSS server machine. **hpssgui** 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** 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** clients for all HPSS administrators and operators. Local desktop **hpssgui** 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** utility to create an SSM user with admin authority. See [The hpssuser utility](#) and the **hpssuser** man page 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** 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** logging in a text file. Bear in mind, however, that this can generate significant amounts of log data. (See the **hpssgui** man page.)
- When you have decided on the **hpssgui** 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** 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** script uses.

## 10.3. 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** or **hpssadm** session. It also explains how to install the SSM client on the user's desktop (the recommended configuration for **hpssgui**) 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** and **hpssadm** 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:

- |                 |   |
|-----------------|---|
| <b>admin</b>    | 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.                               |
| <b>operator</b> | 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** or **hpssadm** on the user's desktop.

See [Using SSM through a firewall](#) for advice about using **hpssgui** or **hpssadm** through a network firewall.

### 10.3.1. 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** or **hpssadm** 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.

## 10.3.2. Creating the SSM user accounts

### 10.3.2.1. The **hpssuser** utility

The **hpssuser** 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** 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** 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.*

### 10.3.2.2. SSM user authorization

SSM user authorization is set properly by the **hpssuser** 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** utility. The **hpssuser** utility program creates and deletes SSM user entries in the AUTHZACL table using the **hpss\_server\_acl** utility. Normally, there is no need to invoke the **hpss\_server\_acl** utility directly because it is invoked by the **hpssuser** utility. However, it is a useful tool for examining and modifying the authorization table.



*Access to the **hpss\_server\_acl** program, **hpssuser** 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** 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.

### 10.3.2.3. 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** or **hpssgui** 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** or **hpssgui** 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** and **hpssadm** 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** utility when the keytab file path is specified. Keytabs may also be created manually with the **hpss\_krb5\_keytab** or **hpss\_unix\_keytab** 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** and **hpssgui** 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** and **hpssgui** 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** utility may be used to generate a keytab with UNIX authentication in the form usable by the **hpssadm** and **hpssgui** programs. See the **hpss\_unix\_keytab** man page for details.

The UNIX keytab is interpreted on the host on which the System Manager runs, not the host on which the **hpssadm** 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** utility must be executed on the host on which the System Manager runs.

The **hpss\_unix\_keytab** 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** 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.

### 10.3.3. SSM configuration file

The **hpssgui** and **hpssadm** scripts use the SSM configuration file **ssm.conf** for configuration.

The **mkhps** utility creates the SSM configuration file for the security mechanism supported by SSM. The **mkhps** 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 **hpssgui** and **hpssadm** script reads. 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 **hpssgui** and **hpssadm** scripts 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 18. SSM general options

File Option	Command Line Option	Functionality
HPSS_SSM_ALARM_RATE	-A	Alarm refresh rate
LOGIN_CONFIG	-C	Full path to <code>login.conf</code> 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 <code>hpssgui/hpssadm</code> waits between polling for lists
HPSS_SSM_ALARMS_DISPLAY	-N	Max number of alarms displayed by <code>hpssgui</code>
HPSS_SSM_CLASSPATH	-P	Full path to <code>hpss.jar</code> 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 <code>krb5.conf</code> file
HPSS_SSM_LANG_CODE	-l	Language code, internationalization
SSM_CONFIG	-m	Full path to SSM configuration file
HPSS_SSM_SM_PORT_NUM	-n	port number <i>or</i> 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 19. HPSSGUI-specific options

File Option	Command Line Option	Functionality
HPSSGUI_AUTHENTICATOR	-a	Authenticator (keytab path name)
HPSSGUI_LOOK_AND_FEEL	-F	Look and feel



File Option	Command Line Option	Functionality
HPSSGUI_MO_RATE	-M	How long <b>hpssgui</b> 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 20. HPSSADM-specific options

File Option	Command Line Option	Functionality
HPSSADM_USER_NAME	-U	User name for <b>hpssadm</b>
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** and **hpssgui** 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.

### 10.3.3.1. login.conf

The **login.conf** file is a login configuration file that specifies the security authentication required for the **hpssgui** and **hpssadm** programs. A copy of the **login.conf** file is included in the **hps.jar** file and should require no site customization. However, a template for the file is provided in **/opt/hps/config/templates/login.conf.template** should the site need to customize the security mechanisms.

See the **/opt/hps/config/templates/login.conf.template** file for details.

### 10.3.3.2. 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.

### 10.3.4. 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**, 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 `/hpss_src/hpss-<version>/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** man page for more details.

### 10.3.5. SSM desktop client packaging

A full installation of HPSS is not needed on machines used only for executing **hpssgui** or **hpssadm**. 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** user. The **hpssgui** 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** 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** 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** 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** utility to package these components. [Manual SSM client packaging and installation](#) describes how to select and package the components manually.

#### 10.3.5.1. Automatic SSM client packaging and installation

The **hpssuser** utility provides a mechanism for packaging all the necessary client files required to execute the **hpssgui** program on the user's desktop host. Refer to the **hpssuser** 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.

#### 10.3.5.2. Manual SSM client packaging and installation

This section describes the manual installation of the necessary client files required to execute the **hpssgui** or **hpssadm** program on the user's desktop host. The **hpssuser** 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**)
- 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      /hpss_src/hpss-<version>/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 AIX, 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** and **hpssadm** man pages for details.

The `hpss.jar` file contains the **hpssadm** and **hpssgui** 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** 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** or **hpssadm** session logs, if these are desired. The session log is an ASCII file that stores messages generated by the **hpssadm** or **hpssgui** programs. By default, the **hpssgui** and **hpssadm** 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** and **hpssadm** 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** or **hpssadm**.

## 10.3.6. Using SSM through a firewall

### 10.3.6.1. The firewall problem

**hpssgui** and **hpssadm** 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:

- **hpssgui** and **hpssadm** 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 **hpssgui** and **hpssadm** also need access to port 111 in order to ask the portmapper where the System Manager is listening.
- If Kerberos authentication is used, then **hpssgui** and **hpssadm** additionally need access to port 88 to communicate with the KDC.

### 10.3.6.2. Solutions for operating through a firewall

SSM can operate through a firewall in three different ways:

- The **hpssgui** and **hpssadm** can use ports exempted by the network administrator as firewall exceptions. See the **-n** option described in the **hpssgui** and **hpssadm** man pages.
- The **hpssgui** and **hpssadm** can contact the System Manager across a Virtual Private Network (VPN) connection. See the **-p** and **-h** options described in the **hpssgui** and **hpssadm** man pages.
- The **hpssgui** and **hpssadm** can contact the System Manager across an SSH tunnel. See the instructions for tunneling in the **hpssgui** 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](#) 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** and **hpssadm**

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.

### 10.3.6.3. Example: Using `hpssgui` through a firewall

Here is an example of how a particular site set up their `hpssgui` 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` 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` 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` script resides. Alternately, specify the file to the `hpssgui` 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` 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` script with the `-P` option, or by using configuration file settings or the appropriate environment variable (see the `hpssgui` 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` script resides. Alternately, specify this file to the `hpssgui` script with the `-k` option. Verify that UDP port 88 on the SSM Server machine is accessible; if not, then `hpssgui` 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.

## 10.4. 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.

## 10.5. 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** 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 within 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,  (filled), 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.

## 10.6. 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.

## 10.7. 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 `"/hpss_src/hpss-<version>/ssmhelp"`. The values of these environment variables may be overridden.

## 10.8. 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.

### 10.8.1. 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.

## 10.8.2. 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.

## 10.8.3. 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.

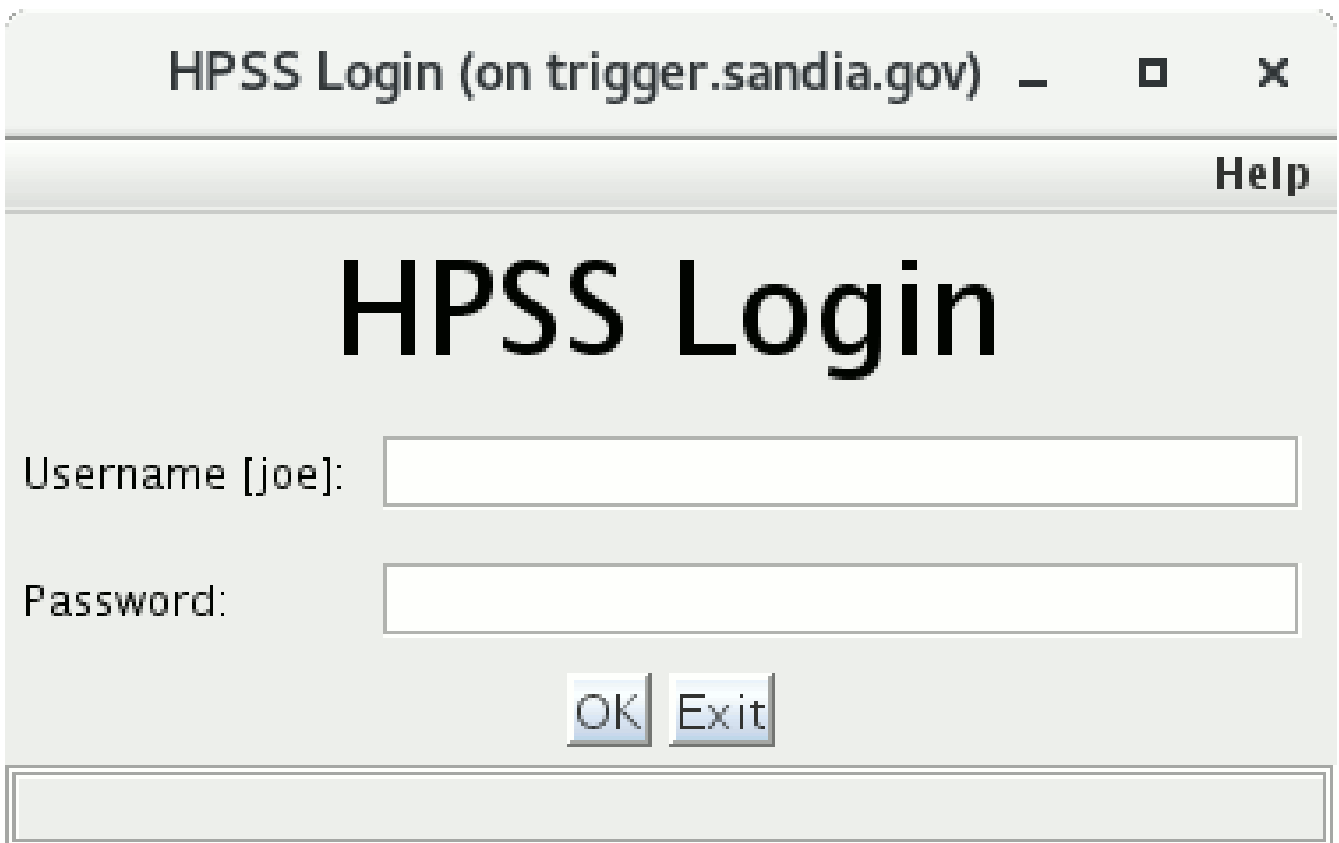
## **List Preferences**

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.

# **10.9. SSM-specific windows**

This section describes the *HPSS Login* window, the *HPSS Health and Status* window, and the SSM information windows.

## **10.9.1. HPSS Login**



The *HPSS Login* window may appear after starting the **hpssgui** 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** or **hpssgui** man pages for more information about the user session log.

#### *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`.

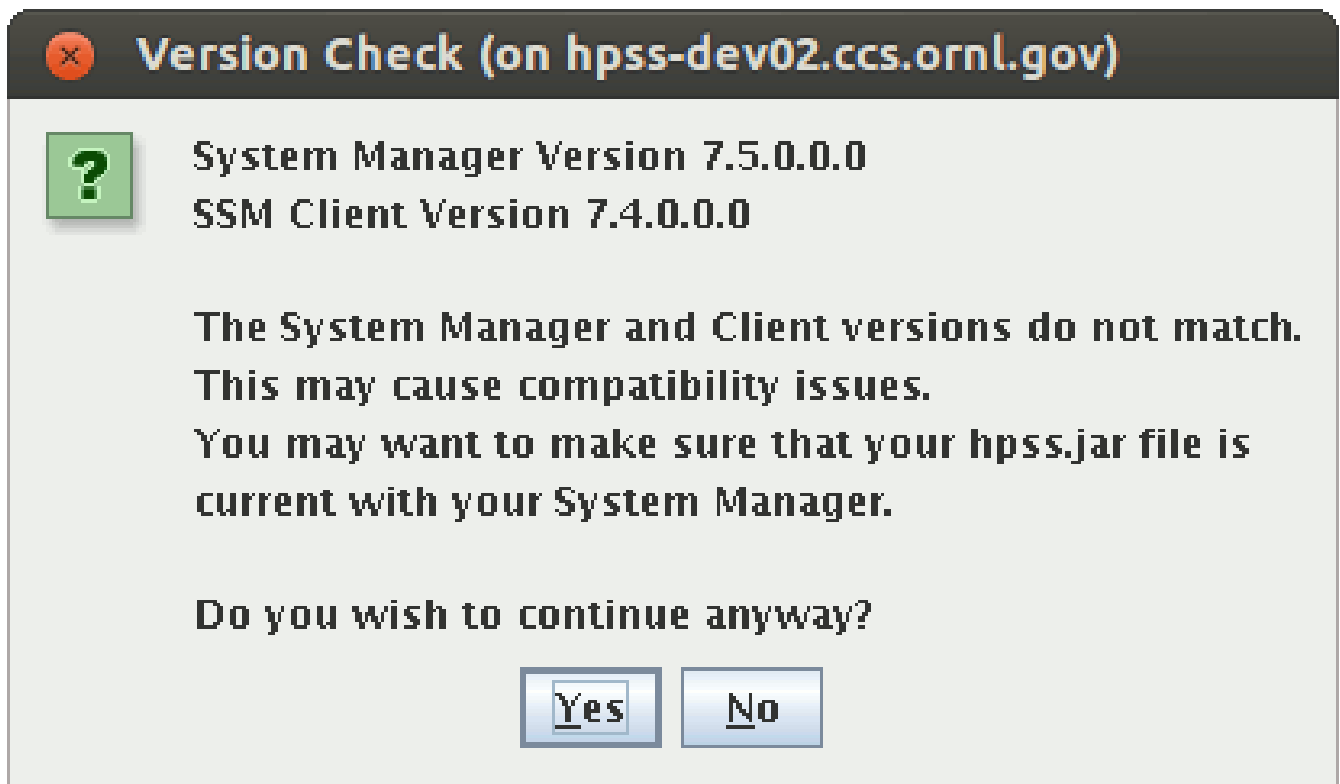
##### **OK**

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** 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.

## 10.9.2. About HPSS



## About HPSS



**HPSS System Name: HPSS System (dus03)**

**System Manager Version 7.5.0.0.0**

**SSM Client Version 7.5.0.0.0**

### **Licensed Materials**

**Copyright (C) 1992, 2018**

**International Business Machines Corporation, The Regents of the University of California, Los Alamos National Security, LLC, Lawrence Livermore National Security, LLC, Sandia Corporation, and UT-Battelle.**

**All rights reserved.**

**Java Version "1.8.0\_161"**

**OpenJDK Runtime Environment (build 1.8.0\_161-b14)**

**OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)**

**Linux (3.10.0-693.17.1.el7.x86\_64 amd64)**

OK

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** 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.

### **10.9.3. HPSS Health and Status**

**HPSS Health and Status**      \_      □      X

---

**File   Edit   View   Monitor   Operations   Configure   Help**

Servers	● Enabled	<input type="button" value="Server List"/>
Devices and Drives	● Disabled	<input type="button" value="Device/Drive List"/>
Storage Class Thresholds	● Critical	<input type="button" value="Storage Class List"/>
PVR Cartridge Thresholds	● OK	<input type="button" value="PVR Information"/>
HPSS Config Changes	● OK	<input type="button" value="Apply Config"/>

Bytes Moved	0
Bytes Used	49,690,245,115
Data Transfers	0
PVL Jobs	0

- Monitor
- Operations
- Configure

Finished applying the HPSS configuration changes      ● SM

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.

### **10.9.3.1. 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.

### **10.9.3.2. 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:

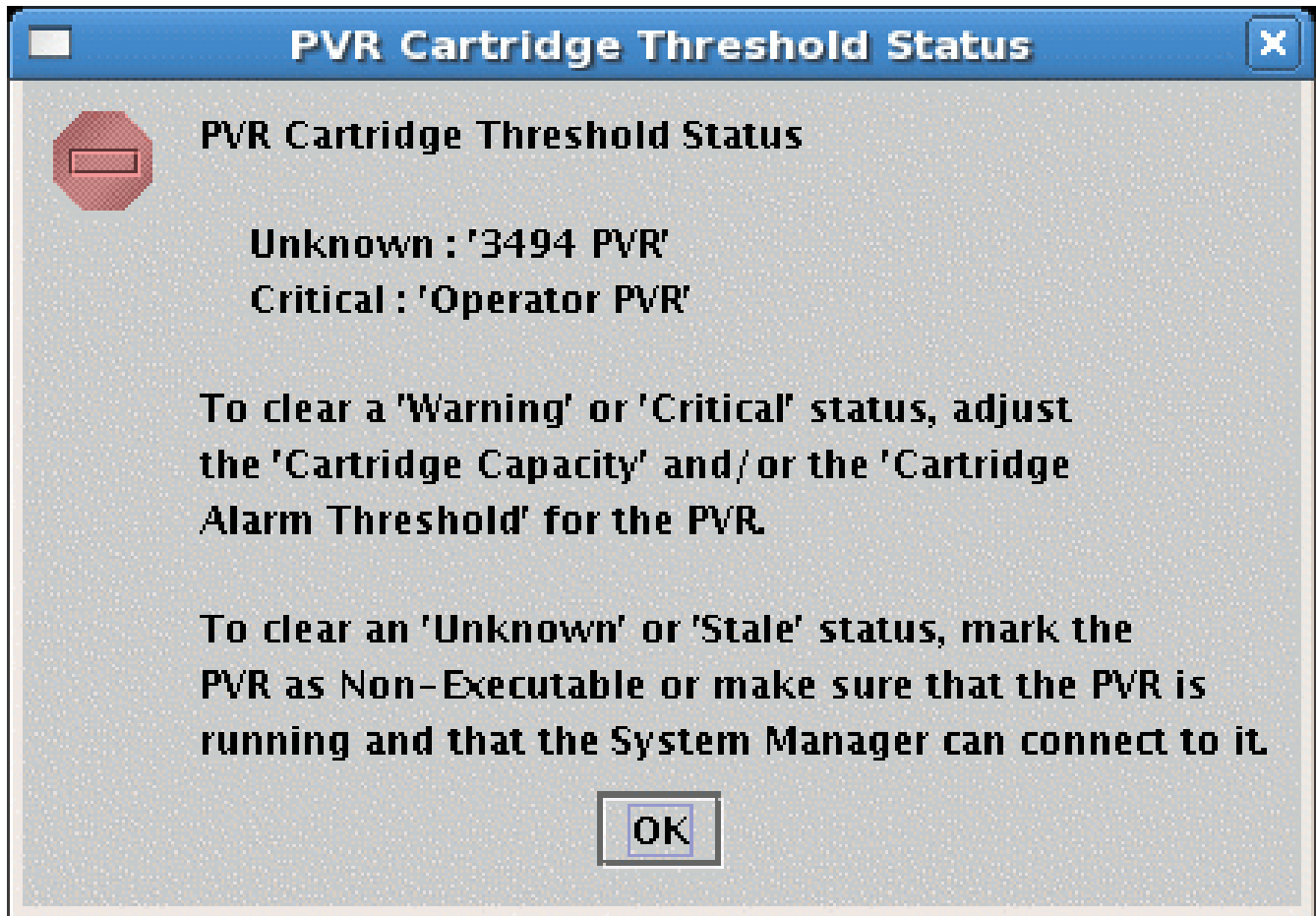
- Red for major and critical problems
- Magenta for minor problems
- Yellow for unknown, stale, suspect, and warning problems
- Green for normal; no problem

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.

### 10.9.3.3. 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.

### 10.9.3.4. 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.

### 10.9.3.5. 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** 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** script will terminate and the user's session log will be closed. The user must rerun the **hpssgui** script to access the *HPSS Login* window and reconnect to the System Manager.

### 10.9.3.6. 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.

## 10.9.4. SSM information windows

These windows describe the System Manager and the user's **hpssgui** session.

### 10.9.4.1. 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**.

**SSM System Manager Statistics**

File Edit Column View Help

Start Time: Apr 16, 2018 5:30:45 PM

Uptime: 00-20:45:53

CPU Time: 00-00:00:21

Memory Usage: 3559120

Process ID: 17937

Hostname: hpss-dev-dus03.ccs.ornl.gov

RPC Calls to Servers: 1788

Alarm Messages: 127

Event Messages: 11764

**RPC Interface Information**

	Server	Client
Status	● OK	● OK
Thread Pool Size	100	100
Request Queue Size	20	600
Active RPCs	0	18
Queued RPCs	0	0
Maximum Active/Queued RPCs	4	20
Data Change Notifications	15960	
Unsolicited Notifications	0	
Tape Check-In Messages	0	
Tape Mount Messages	266	
Total RPCs	16226	53010

**Client Connection Information**

Maximum Connections: 4

Current Connection Count: 4

Current Client Count: 2

ID	State	User Auth	Principal	Hostname
0	Free	admin	dusadmin@hpss-dev-dus03.ccs.ornl.gov	hpss-dev-dus03.ccs.ornl.gov
1	Free	admin	dusadmin@hpss-dev-dus03.ccs.ornl.gov	hpss-dev-dus03.ccs.ornl.gov
2	Free	admin	dusadmin@hpss-dev-dus03.ccs.ornl.gov	hpss-dev-dus03.ccs.ornl.gov
3	Free	admin	dusadmin@hpss-dev-dus03.ccs.ornl.gov	hpss-dev-dus03.ccs.ornl.gov
4	Free	admin	dusadmin@hpss-dev-dus03.ccs.ornl.gov	hpss-dev-dus03.ccs.ornl.gov
5	Free	admin	dusadmin@hpss-dev-dus03.ccs.ornl.gov	hpss-dev-dus03.ccs.ornl.gov
6	Free	admin	dusadmin@hpss-dev-dus03.ccs.ornl.gov	hpss-dev-dus03.ccs.ornl.gov

16/16

Freeze Refresh Dismiss

**Preferences**

Edit

Default

### 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 **hpssgui** and **hpssadm** programs 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.

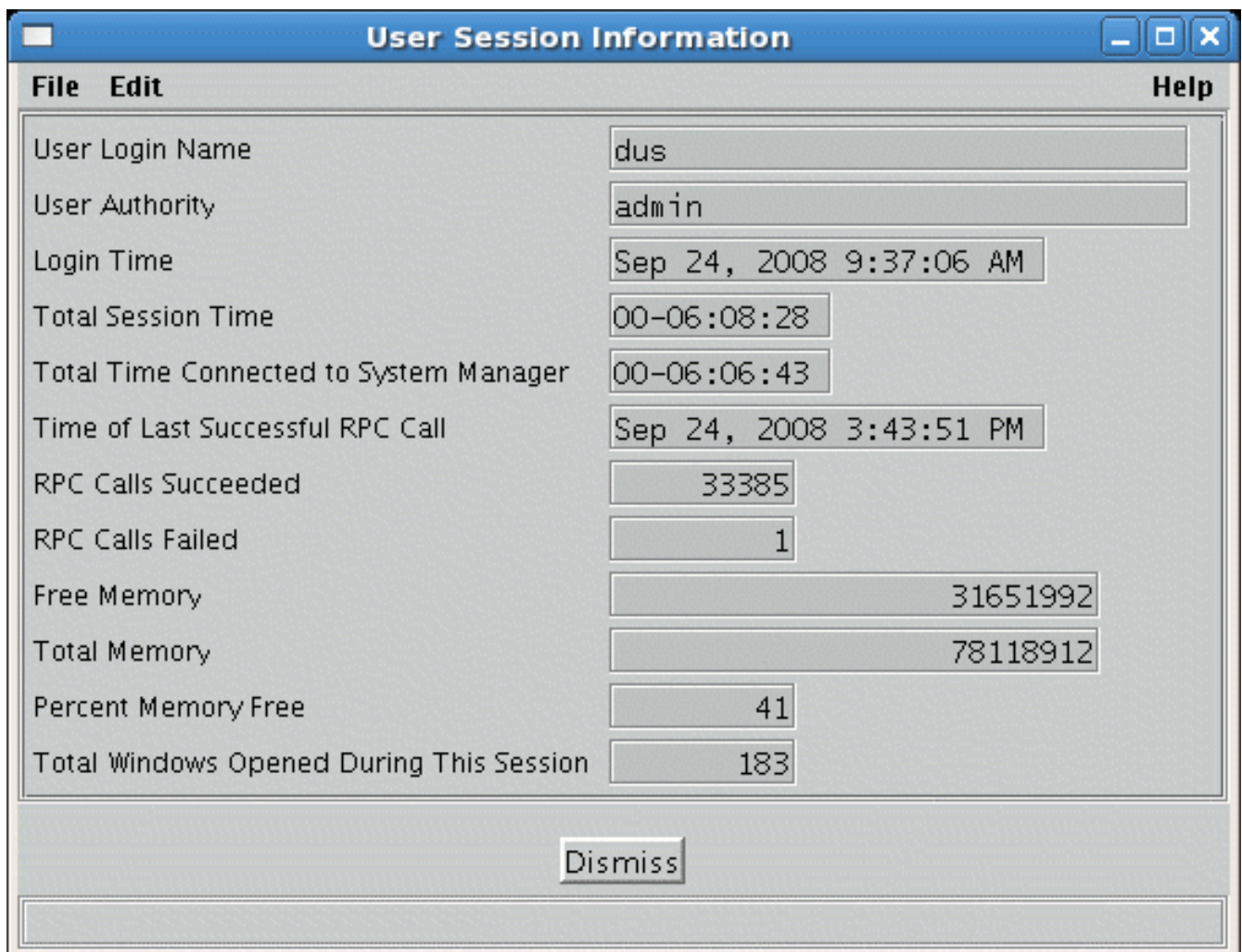
### Client UUID

The UUID for the client.

#### 10.9.4.2. 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** program.

### **Total Memory**

Amount of total memory available in the **hpssgui** program.

### **Percent Memory Free**

The ratio of free memory to total memory in the **hpssgui** process.

### **Total Windows Opened During This Session**

The number of windows created during the current user session.

## 10.10. 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** 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 11. 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 \(new HPSS sites\)](#).

## 11.1. Global Configuration window

Global Configuration
Help

File
Edit

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.*

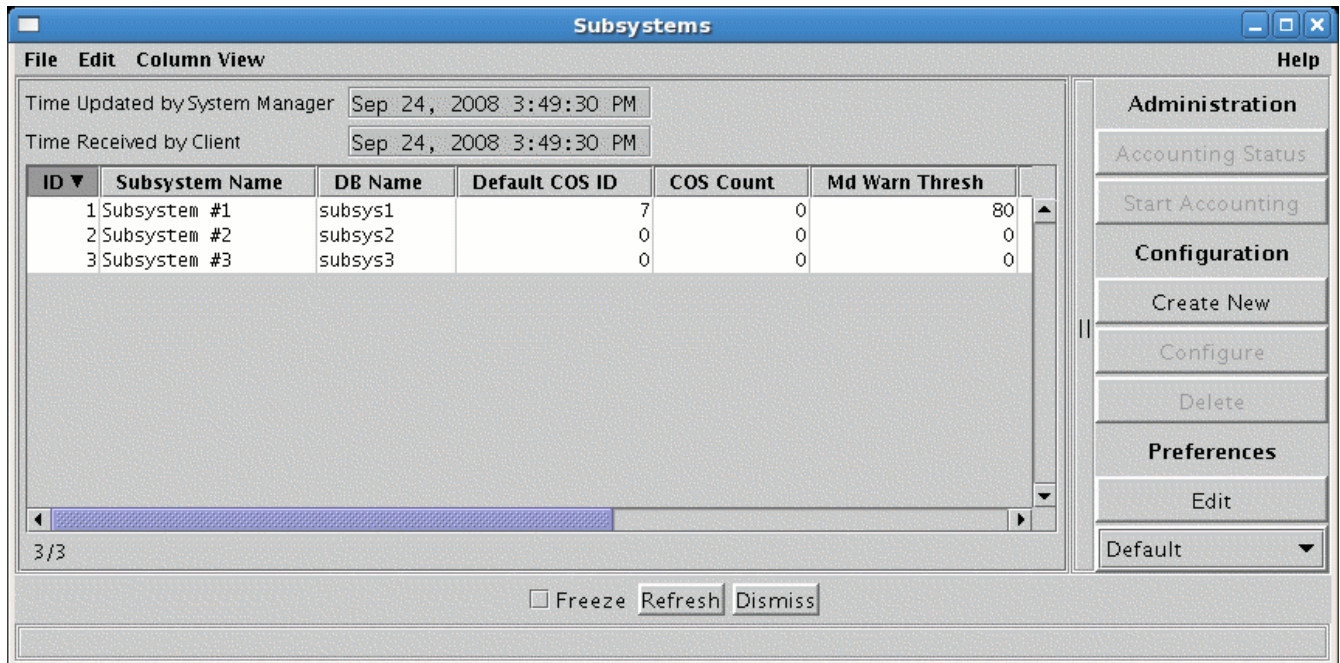
*When changes are made to trashcan settings, the Core Servers must be reinitialized or restarted for the new settings to take effect. It is recommended to use the **Apply Config** button on the*

## 11.2. 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.

### 11.2.1. 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](#)

## 11.2.2. 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](#)

### 11.2.3. Storage Subsystem Configuration window

ID	COS Name	Default COS	Allow
1	Class of Service 1	yes	<input checked="" type="checkbox"/>
2	Class of Service 2 (Tape)		<input checked="" type="checkbox"/>
3	Class of Service 3		<input checked="" type="checkbox"/>
4	Class of Service 4		<input checked="" type="checkbox"/>

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.

### 11.2.3.1. 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](#)

### 11.2.3.2. 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.

### 11.2.3.3. 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.

### 11.2.3.4. 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.

### 11.2.3.5. 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.

### 11.2.3.6. 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.

### 11.2.3.7. 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.

### 11.2.3.8. 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.

## 11.2.4. 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.

## 11.2.5. 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** 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 nsubject
```

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.

## 11.3. 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.

### 11.3.1. 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.

### 11.3.2. 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.

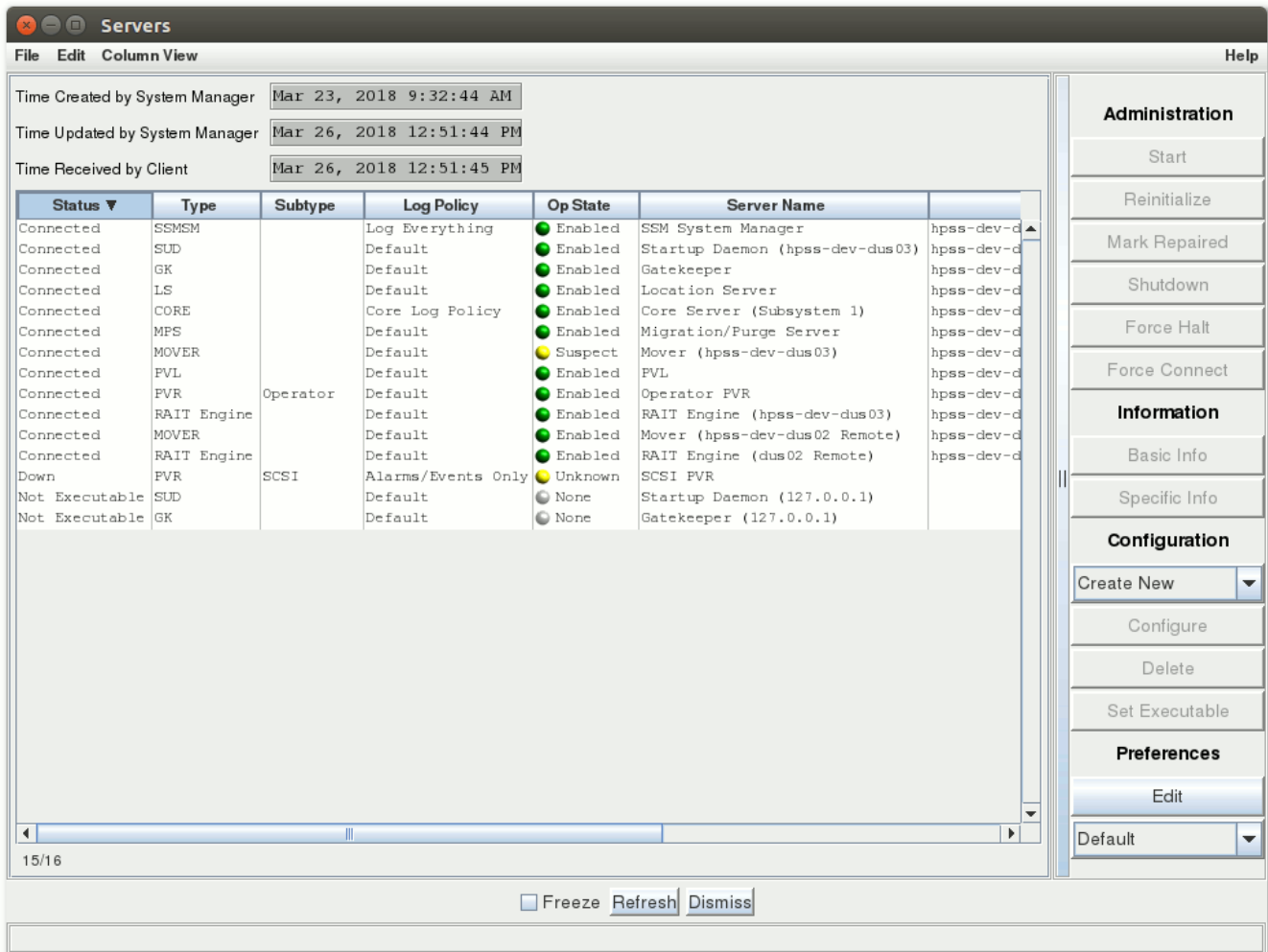
## Trash Statistics Interval

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 12. 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.

## 12.1. 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** or **hpssadm** 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** or **hpssadm** 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.

## 12.2. 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.

## 12.2.1. Common server configuration

These sections of the *Server Configuration* window are common to all servers.

### 12.2.1.1. Basic controls

The screenshot shows a window titled "Core Server Configuration" with a menu bar containing "File", "Edit", and "Help". The window is divided into two main sections: "Basic Controls" and "Execution Controls".

**Basic Controls:**

- Server Name: Core Server (Subsystem 1)
- Server ID: 7
- UUID: 4ca0aad4-8e19-11e5-83b3-005056b7004f
- Server Type: CORE
- Storage Subsystem: Subsystem #1 (dropdown)
- Log Policy: Core Log Policy (dropdown)

**Execution Controls:**

- Execute Pathname: /opt/hpss/bin/hpss\_core
- Execute Hostname: hpss-dev-dus03.ccs.ornl.gov
- Program Number: 536870917
- Version Number: 1
- UNIX Username: root
- Auto Restart Count: 3
- Executable
- Auto Startup

At the bottom of the window, there are buttons for "Update", "Delete", "Start Over", "Clone (partial)", and "Dismiss".

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.



### 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.

#### 12.2.1.2. Execution controls

The screenshot shows a window titled "Core Server Configuration" with a menu bar containing "File", "Edit", and "Help". The window is divided into several sections:

- 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 (Active Tab):**
  - Execute Pathname: /opt/hpss/bin/hpss\_core
  - Execute Hostname: hpss-dev-dus03.ccs.ornl.gov
  - Program Number: 536870917
  - Version Number: 1
  - UNIX Username: root
  - Auto Restart Count: 3
  - Executable
  - Auto Startup
- Other Tabs:** Interface Controls, Security Controls, Audit Policy, Specific
- Buttons:** Update, Delete, Start Over, Clone (partial), Dismiss

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.

#### **12.2.1.3. 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.

#### 12.2.1.4. Security controls

The security controls fields define the settings for authenticated communication between the server and other HPSS servers and clients.

The screenshot shows the 'Core Server Configuration' window with the 'Security Controls' tab selected. The window has a menu bar with 'File', 'Edit', and 'Help'. The 'Basic Controls' section includes fields for Server Name (Core Server (Subsystem 1)), Server ID (7), UUID (4ca0aad4-8e19-11e5-83b3-005056b7004f), Server Type (CORE), Storage Subsystem (Subsystem #1), and Log Policy (Core Log Policy). The 'Security Controls' section includes Principal Name (hpsscore), Protection Level (Connect), and two 'Authentication Service Configuration' sections. The first section has Mechanism (UNIX), Authenticator Type (Keytab), and Authenticator (auth\_keytab:/var/hpss/etc/hpss.unix.keytab). The second section has Mechanism (KRB5), Authenticator Type (Keytab), and Authenticator (auth\_keytab:/var/hpss/etc/hpss.keytab). At the bottom, there are buttons for Update, Delete, Start Over, Clone (partial), and Dismiss.

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** 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.

#### **12.2.1.5. Audit policy**

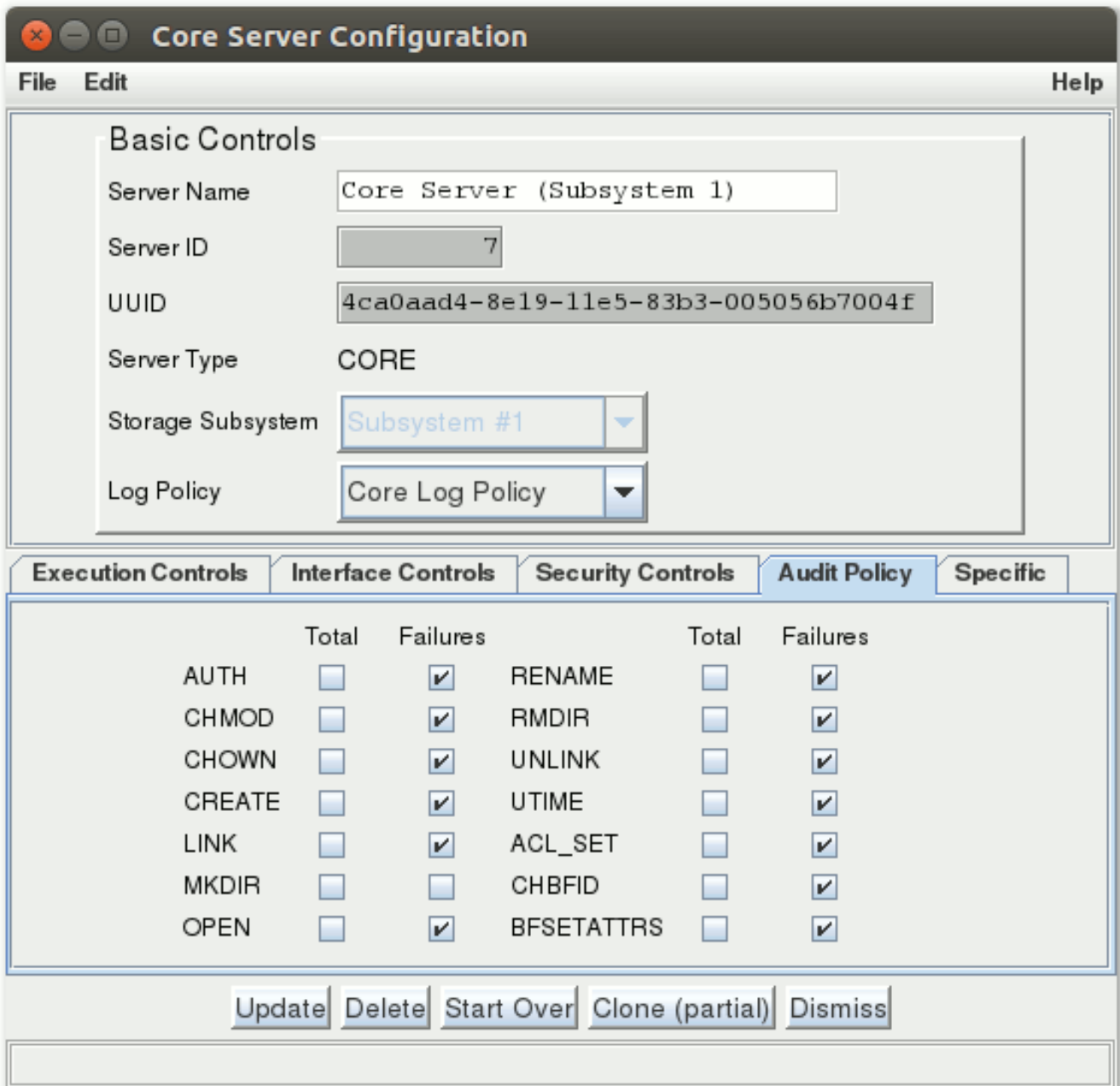
HPSS security provides a log for certain security-related events such as a change in an object's permissions, a change in an object's owner, a change of an object's name, a failed attempt to access an object, and several others (see *Field descriptions* below for a complete list). A server's Audit Policy 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**

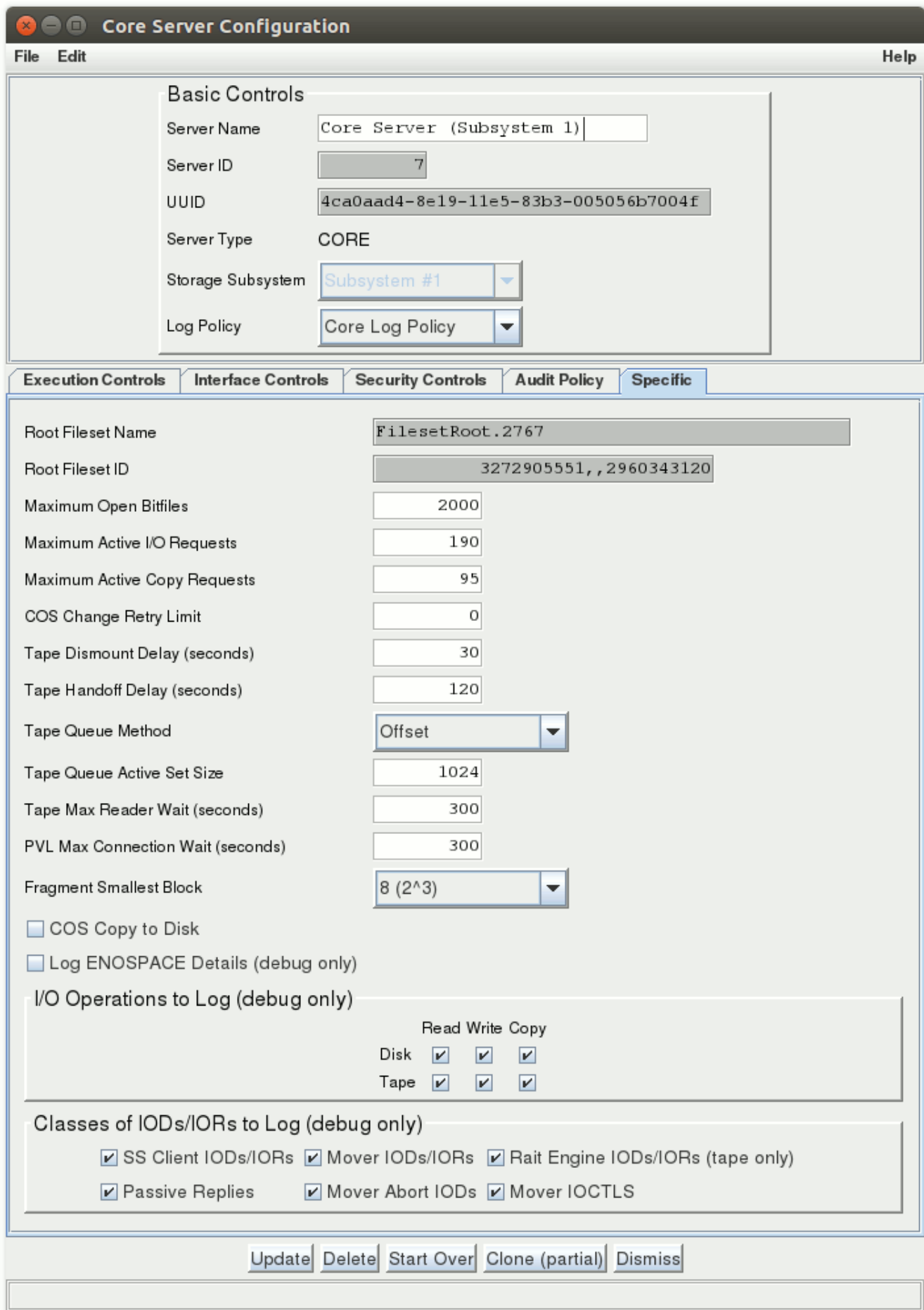
Core Server change bitfile identifier events.

**BFSETATTRS**

Core Server set bitfile attribute events.

**12.2.2. 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.

#### **12.2.2.1. 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\_ETIME\_POLICY**

Selects the algorithm the Core Server will use whenever it computes the etime. There are three possible algorithms: CORE\_ETIME\_POLICY\_NOETIME, CORE\_ETIME\_POLICY\_RELATIVE, 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 traditional 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** will be written to tapes that contain only files written by the **repack**. If this environment variable has the value "off", files moved from one tape virtual volume to another by **repack** 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** 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** 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** 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.



#### *Advice*

\_For more information on tape ordered recall, see [Overview of TOR](#)

#### 12.2.2.2. 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:

$$\text{percentage} = \frac{(\text{configured Max Active Copy IO})}{(\text{configured Max Active Total IO})}$$

3. It sets its actual Maximum Active Copy IO to this percentage of its actual Maximum Active Total IO:

$$\text{Actual Max Active Copy IO} = \text{percentage} * \text{Actual Max Active Total IO}$$

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 scrub limits command, or issue a SIGHUP to the Core Server.

Here is an example of the scrub limits command:

```
scrub> limits
      Max Open Bitfiles =          14000
      Cur Open Bitfiles =           12
      Max Active IO =             600
      Cur Active IO =              0
      Max Active Copy IO =         600
      Cur Active Copy IO =         0
      Max Connections =           1600
      Cur Connections =            3
      Tape Queue Method =         Offset
```

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.

### 12.2.2.3. 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:

$$50,000 * \text{stream count} * 2 * \text{copy count}$$

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)
```

### 12.2.3. Gatekeeper-specific configuration

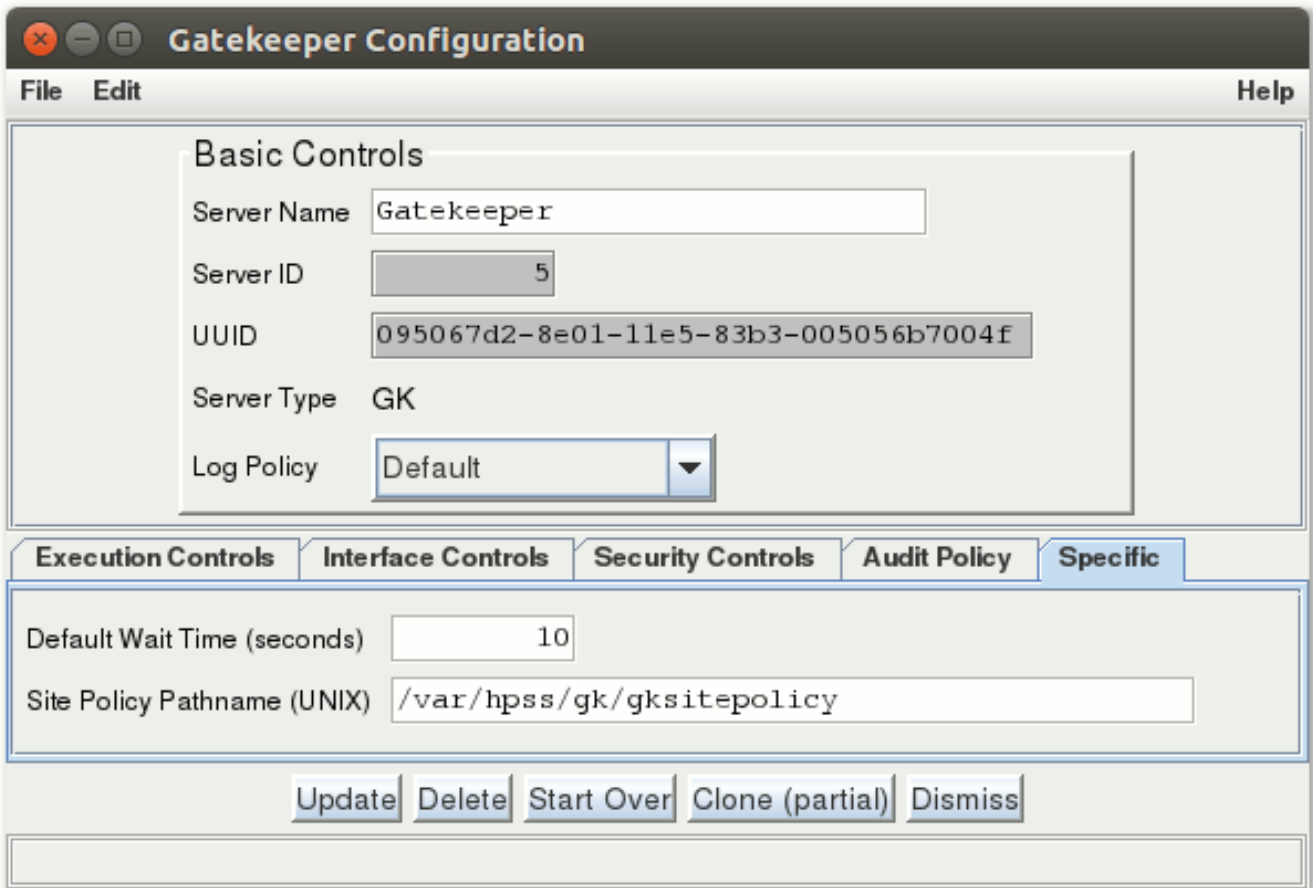
To use a Gatekeeper for gatekeeping services the Gatekeeper must be configured into one or more storage subsystems (see [Storage Subsystem Configuration window](#)). To associate a Gatekeeper with a storage subsystem, the Gatekeeper must be selected on the *Storage Subsystem Configuration* window.

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.

## 12.2.4. 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`.

### 12.2.5. Migration/Purge Server (MPS)-specific configuration

The screenshot shows a window titled "Migration/Purge Server Configuration" with a menu bar (File, Edit, Help). The "Basic Controls" section includes fields for Server Name (Migration/Purge Server), Server ID (9), UUID (a333e356-8e1a-11e5-83b3-005056b7004f), Server Type (MPS), Storage Subsystem (Subsystem #1), and Log Policy (Default). Below this is a tabbed interface with tabs for Execution Controls, Interface Controls, Security Controls, Audit Policy, and Specific. The "Specific" tab is active, showing "Storage Class Update Interval" (60 seconds), "Core Server API Failures Allowed" (3), and "MPS Report File Name (path/prefix)". At the bottom are buttons for Update, Delete, Start Over, Clone (partial), and Dismiss.

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.

## **12.2.6. Mover-specific configuration**

### **12.2.6.1. Mover-specific configuration window**

✕ ◀ ◻
Mover Configuration

File Edit
Help

### Basic Controls

Server Name

Server ID

UUID

Server Type **MOVER**

Log Policy  ▼

Execution Controls
Interface Controls
Security Controls
Audit Policy
Specific

Encryption Key

Buffer Size

TCP Path Name

Hostname

TCP Port

Port Range Start

Port Range End

#### Data Hostnames

0	<input type="text" value="hpss-dev-dus03.ccs.ornl.gov"/>
1	<input type="text"/>
2	<input type="text"/>
3	<input type="text"/>
4	<input type="text"/>
5	<input type="text"/>
6	<input type="text"/>
7	<input type="text"/>



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 21. 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.

#### 12.2.6.2. 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.tpl
/opt/hpss/config/templates/system_files/hpssmvr@.service.systemd.tpl
```

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.tpl \
/etc/systemd/system/hpssmvr.socket
cp /opt/hpss/config/templates/system_files/hpssmvr@.service.systemd.tpl \
/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 either the `/etc/inetd.conf` or `/etc/xinetd.d` file. For example, if the encryption key in the Mover's type-specific configuration is `1234567890ABCDEF`, then the encryption key file (`/var/hpss/etc/ek.mvr1`) should contain:

```
0x12345678 0x90ABCDEF
```

### `/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 **mkhps** 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 **mkhps** 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 **mkhps** 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](#). At this time, there are no additional system parameters to set for AIX, so it is not actually necessary to select this check box for AIX hosts.

### **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/hps/et/hps_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.*

## 12.2.7. 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.*

### 12.2.7.1. Operator PVR-specific configuration window

**Operator PVR Configuration**

File Edit Help

**Basic Controls**

Server Name: Operator PVR

Server ID: 12

UUID: 8bcc78a6-8e1d-11e5-83b3-005056b7004f

Server Type: PVR

Server Subtype: Operator

Log Policy: Default

Execution Controls | Interface Controls | Security Controls | Audit Policy | **Specific**

Cartridge Capacity: 1000

Cartridge Alarm Threshold: 80 percent

Shelf Tape Check-In Retry: 30 seconds

Shelf Tape Check-In Alarm: 10 minutes

Dismount Delay: 15 minutes

Defer Dismount Exempt Count: -1

**Characteristics**

Defer Dismounts  Support Shelf Tape

Update Delete Start Over Clone (partial) Dismiss

### 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** utility.

### 12.2.7.2. SCSI PVR-specific configuration window

**SCSI PVR Configuration**

File Edit Help

### Basic Controls

Server Name

Server ID

UUID

Server Type PVR

Server Subtype SCSI

Log Policy  ▼

---

**Execution Controls** | **Interface Controls** | **Security Controls** | **Audit Policy** | **Specific**

Cartridge Capacity

Cartridge Alarm Threshold  percent

Same Job on Controller

Other Job on Controller

Distance To Drive

Shelf Tape Check-In Retry  seconds

Shelf Tape Check-In Alarm  minutes

Dismount Delay  minutes

Retry Mount Time Limit  minutes

Drive Error Limit

Defer Dismount Exempt Count

### Characteristics

Defer Dismounts  Support Shelf Tape  Enforce Home Location

Serial Number 0

Serial Number 1

Serial Number 2

Serial Number 3

Serial Number 4

Serial Number 5

Serial Number 6

Serial Number 7

### 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:

$$\begin{aligned} \text{Score} = & \\ & \text{Weight 1} \times \text{Cartridges from this job mounted on this drive's controller} + \\ & \text{Weight 2} \times \text{Cartridges from other jobs mounted on this drive's controller} \\ & + \\ & \text{Weight 3} \times \text{Units of distance from the cartridge to the drive} \end{aligned}$$

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** 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** utility can be used to view the home location values.

### Serial Number 0 - 7

The serial number of the robots, obtained from **device\_scan**. 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** 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 = [B071]
   Vendor info = [B071780000078A3467 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.

### 12.2.7.3. STK PVR-specific configuration window

**STK PVR Configuration**

File Edit Help

**Basic Controls**

Server Name: STK PVR

Server ID: 175

UUID: 284c0bf8-311c-11e8-ad3c-005056b7004f

Server Type: PVR

Server Subtype: STK

Log Policy: Default

---

**Execution Controls** | **Interface Controls** | **Security Controls** | **Audit Policy** | **Specific**

Cartridge Capacity: 6000

Cartridge Alarm Threshold: 90 percent

Same Job on Controller: 10

Other Job on Controller: 5

Distance To Drive: 20

Shelf Tape Check-In Retry: 30 seconds

Shelf Tape Check-In Alarm: 10 minutes

Dismount Delay: 15 minutes

Retry Mount Time Limit: -1 minutes

Drive Error Limit: -1

Defer Dismount Exempt Count: -1

**Characteristics**

Defer Dismounts  Support Shelf Tape

ACSLs Packet Version: 4

Alternate SSI Port:

**Update** **Delete** **Start Over** **Clone (partial)** **Dismiss**

Successfully saved the 'STK PVR Configuration'

*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:

$$\begin{aligned} \text{Score} = & \\ & \text{Weight 1} \times \text{Cartridges from this job mounted on this drive's controller} + \\ & \text{Weight 2} \times \text{Cartridges from other jobs mounted on this drive's controller} \\ & + \\ & \text{Weight 3} \times \text{Units of distance from the cartridge to the drive} \end{aligned}$$

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** 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 AIX, 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*

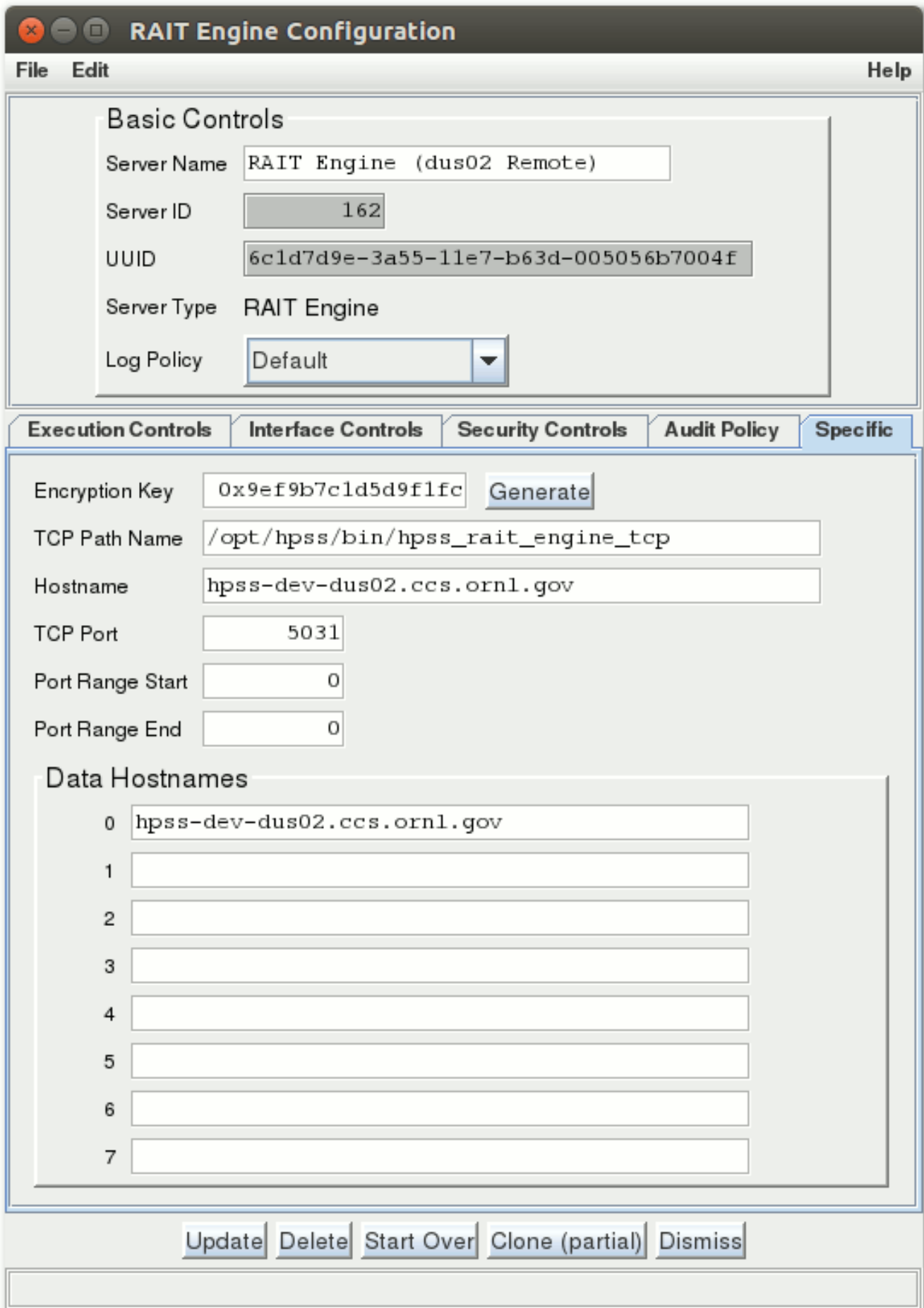
```

## 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.

## **12.2.8. RAIT Engine-specific configuration**

### **12.2.8.1. 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.tpl  
/opt/hpss/config/templates/system_files/hpssrait@.service.systemd.tpl
```

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.tpl \  
/etc/systemd/system/hpssrait.socket  
cp /opt/hpss/config/templates/system_files/hpssrait@.service.systemd.tpl \  
/etc/systemd/system/hpssrait@.service
```

Once the configuration files have been modified to have the desired settings, then the enable the socket.

```
systemctl enable rait.socket  
systemctl start rait.socket
```

Check that the socket is enable and active

```
systemctl status rait.socket
```

## 12.2.9. 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.*

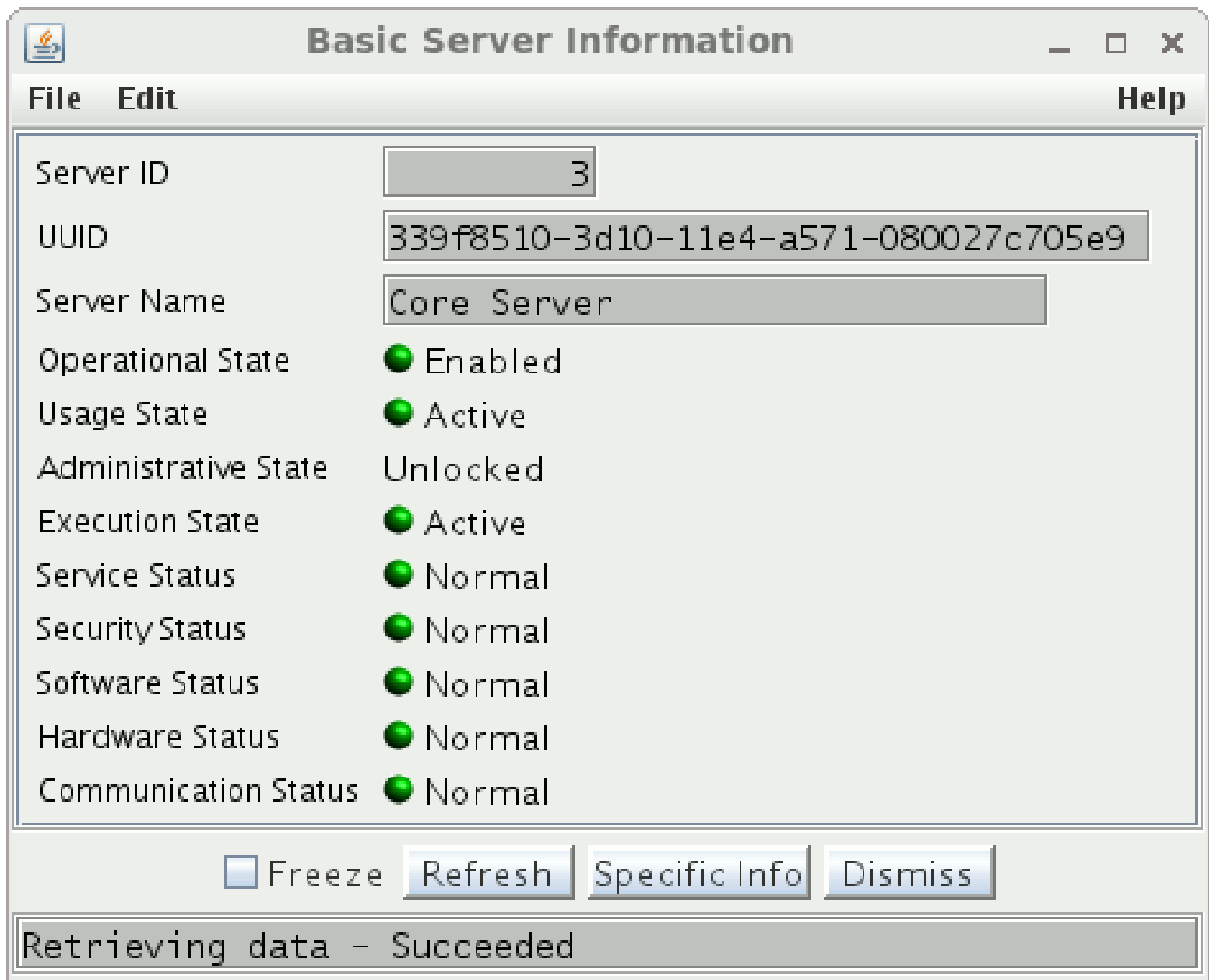
- If deleting a PVR configuration, update the **PVR** field in the appropriate device and drive configuration using [Devices and Drives window](#).
  - If deleting a Mover configuration, update the Mover field in the appropriate device and drive configuration using [Devices and Drives window](#).
3. 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.

## 12.3. 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.

### 12.3.1. 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 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.*

## 12.3.2. 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.

### 12.3.2.1. Core Server Information window

**Core Server Information**

File Edit Help

Core Server: Core Server

Global Database Name: cfgx

Subsystem Database Name: subsysx1

Schema Name: HPSS

Root Fileset Name: FilesetRoot.449

Root Fileset ID: 3272831213, ,3838938124

Maximum Open Bitfiles: 2000

Maximum Active I/O Reqs: 190

Maximum Active Copy Reqs: 95

COS Change Stream Count: 1

COS Change Retry Limit: 0

Tape Dismount Delay (seconds): 30

Tape Handoff Delay (seconds): 120

Tape Queue Method: Offset

Tape Queue Active Set Size: 1024

Tape Max Reader Wait (seconds): 300

PVL Max Connection Wait (seconds): 300

Fragment Smallest Block: 8 (2^3)

Log ENOSPACE Details (debug only)

Root User ID: 0  Root is Superuser

I/O Operations to Log (debug only)

Read Write Copy

Disk

Tape

Classes of IODs/IORs to Log (debug only)

SS Client IODs/IORs  Mover IODs/IORs  Rait Engine IODs/IORs (tape only)

Passive Replies  Mover Abort IODs  Mover IOCTLS

**Name Space Statistics** **Disk/Tape Statistics** **Trashcan Statistics** **Options**

**Subsystem Statistics**

Stages: 0

Migrations: 0

Purges: 0

File Deletes: 0

Last Reset Time: Jan 26, 2015 4:53:04 PM

Freeze

Retrieving data - Succeeded

*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.

**Core Server Information**

File Edit Help

Core Server: Core Server

Global Database Name: cfgx

Subsystem Database Name: subsysx1

Schema Name: HPSS

Root Fileset Name: FilesetRoot.449

Root Fileset ID: 3272831213, 3838938124

Maximum Open Bitfiles: 2000

Maximum Active I/O Reqs: 190

Maximum Active Copy Reqs: 95

COS Change Stream Count: 1

COS Change Retry Limit: 0

Tape Dismount Delay (seconds): 30

Tape Handoff Delay (seconds): 120

Tape Queue Method: Offset

Tape Queue Active Set Size: 1024

Tape Max Reader Wait (seconds): 300

PVL Max Connection Wait (seconds): 300

Fragment Smallest Block: 8 (2^3)

Log ENOSPACE Details (debug only)

Root User ID: 0  Root is Superuser

I/O Operations to Log (debug only)

Read Write Copy

Disk

Tape

Classes of IODs/IORs to Log (debug only)

SS Client IODs/IORs  Mover IODs/IORs  Rait Engine IODs/IORs (tape only)

Passive Replies  Mover Abort IODs  Mover IOCTLS

**Name Space Statistics** | **Disk/Tape Statistics** | **Trashcan Statistics** | **Options**

**Subsystem Statistics**

Files	33	Hard Links	0
Directories	23	Junctions	13
Symbolic Links	1	Filesets	2
		Total Name Space Objects	72

Freeze

Retrieving data - Succeeded

## 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.

**Core Server Information**

**File Edit** **Help**

---

Core Server: Core Server

Global Database Name: cfg

Subsystem Database Name: subsys1

Schema Name: HPSS

Root Fileset Name: FilesetRoot.3539

Root Fileset ID: 3272984902,,2667178584

Maximum Open Bitfiles: 200000

Maximum Active I/O Reqs: 80

Maximum Active Copy Reqs: 80

COS Change Stream Count: 1

COS Change Retry Limit: 0

Tape Dismount Delay (seconds): 600

Tape Handoff Delay (seconds): 300

Tape Queue Method: Offset

Tape Queue Active Set Size: 1024

Tape Max Reader Wait (seconds): 300

PVL Max Connection Wait (seconds): 300

Fragment Smallest Block: 1 (2^0)

Log ENOSPACE Details (debug only)

Root User ID: 0  Root is Superuser

I/O Operations to Log (debug only)

	Read	Write	Copy
Disk	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Tape	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Classes of IODs/IORs to Log (debug only)

SS Client IODs/IORs  Mover IODs/IORs  Rait Engine IODs/IORs (tape only)

Passive Replies  Mover Abort IODs  Mover IOCTLS

**Subsystem Statistics** | **Name Space Statistics** | **Disk/Tape Statistics** | **Trashcan Statistics** | **Options**

Total Disk Virtual Volumes	3	Active Disk Bytes	62,019,076,096
Total Tape Virtual Volumes	38	Free Disk Bytes	2,137,105,891,328
Tape Aggregates	1,881	Active Tape Bytes	15,347,628,285
File Segments in Tape Aggregates	2,034,598	Free Tape Bytes	44,667,659,878,400
Bytes in Tape Aggregates	4,301,097,060		

Freeze

Retrieving data - Succeeded

## 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.

Core Server Information
⌵ □ ✕

File Edit
Help

Core Server	Core Server
Global Database Name	cfgx
Subsystem Database Name	subsysx1
Schema Name	HPSS
Root Fileset Name	FilesetRoot.449
Root Fileset ID	3272831213, ,3838938124
Maximum Open Bitfiles	2000
Maximum Active I/O Reqs	190
Maximum Active Copy Reqs	95
COS Change Stream Count	1
COS Change Retry Limit	0
Tape Dismount Delay (seconds)	30
Tape Handoff Delay (seconds)	120
Tape Queue Method	Offset
Tape Queue Active Set Size	1024
Tape Max Reader Wait (seconds)	300
PVL Max Connection Wait (seconds)	300
Fragment Smallest Block	8 (2^3)

Log ENOSPACE Details (debug only)

Root User ID   Root is Superuser

**I/O Operations to Log (debug only)**

	Read	Write	Copy
Disk	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tape	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Classes of IODs/IORs to Log (debug only)**

SS Client IODs/IORs    Mover IODs/IORs    Rait Engine IODs/IORs (tape only)

Passive Replies    Mover Abort IODs    Mover IOCTLS

Name Space Statistics
Disk/Tape Statistics
Trashcan Statistics
Options

**Subsystem Statistics**

Last Trash Statistics Time	Jan 27, 2015 11:45:19 AM
Total Objects in the Trash	0
Size of Objects in the Trash	0
Total Objects Eligible for Unlink	0
Size of Objects Eligible for Unlink	0

Freeze

Retrieving data - Succeeded

## 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.



**Core Server Information**

File Edit Help

Core Server: Core Server

Global Database Name: cfgx

Subsystem Database Name: subsysx1

Schema Name: HPSS

Root Fileset Name: FilesetRoot.449

Root Fileset ID: 3272831213, ,3838938124

Maximum Open Bitfiles: 2000

Maximum Active I/O Reqs: 190

Maximum Active Copy Reqs: 95

COS Change Stream Count: 1

COS Change Retry Limit: 0

Tape Dismount Delay (seconds): 30

Tape Handoff Delay (seconds): 120

Tape Queue Method: Offset

Tape Queue Active Set Size: 1024

Tape Max Reader Wait (seconds): 300

PVL Max Connection Wait (seconds): 300

Fragment Smallest Block: 8 (2^3)

Log ENOSPACE Details (debug only)

Root User ID: 0  Root is Superuser

**I/O Operations to Log (debug only)**

	Read	Write	Copy
Disk	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tape	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Classes of IODs/IORs to Log (debug only)**

SS Client IODs/IORs  Mover IODs/IORs  Rait Engine IODs/IORs (tape only)

Passive Replies  Mover Abort IODs  Mover IOCTLS

**Name Space Statistics** | **Disk/Tape Statistics** | **Trashcan Statistics** | **Options**

**Subsystem Statistics**

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

COS Copy to Disk

Account Flag Switch

Freeze

Retrieving data - Succeeded

## 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.

## **12.3.2.2. Gatekeeper Information window**

Gatekeeper Information

File Edit Help

Server Name: Gatekeeper

Default Wait Time (seconds): 10

Site Policy Pathname (UNIX): /var/hpss/gk/gksitepolicy

Administrative Activity Statistics

	Calls	Errors	Retries
Get Basic Server Info	0	0	
Set Basic Server Info	2	0	
Get Gatekeeper Server Info	0	0	
Set Gatekeeper Server Info	4	0	
Creates	0	0	0
Auth Caller Creates	0	0	
Creates Completed	0	0	
Opens	0	0	0
Auth Caller Opens	0	0	
Closes	0	0	
Stages	0	0	0
Auth Caller Stages	0	0	
Stages Completed	0	0	
Client Clean Ups	0	0	
Get Monitor Types	0	0	
Pass Thrus	0	0	
Queries	0	0	
Read Site Policies	0	0	

Jan 28, 2013 2:48:11 PM Last Reset Time

Reset

Read Site Policy

Freeze Update Refresh Dismiss

Retrieving data - Succeeded

### 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.

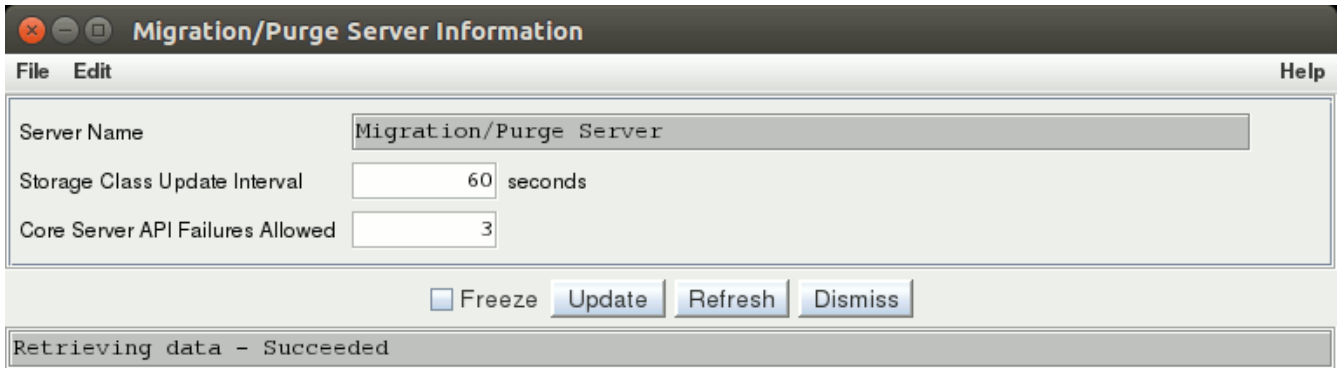
Note: a "policy change" cannot include a change in the types of requests being monitored. When changing the types of requests being monitored (authorized caller, create, open, and stage), the Gatekeeper must be recycled so that the Core Server can learn which request types need to

include Gatekeeping services.

### 12.3.2.3. 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.

### 12.3.2.4. Mover Information window

The screenshot shows a window titled "Mover Information" with a menu bar containing "File", "Edit", and "Help". The main area contains several fields and buttons:

- Server Name: Mover (beaker42)
- Number Of Request Tasks: 1
- Number Of Active Requests: 0
- Time Of Last Statistics Reset: Sep 18, 2008 9:38:20 AM
- Buffer Size: 1048576
- Number Of Requests Processed: 29 (with a "Reset" button)
- Number Of Request Errors: 6 (with a "Reset" button)
- Number Of Data Transfers: 0 (with a "Reset" button)
- Number Of Bytes Moved: 0 (with a "Reset" button)
- Total Move Time: 00-00:00:00 (with a "Reset" button)

At the bottom, there is a "Freeze" checkbox, and "Update", "Refresh", and "Dismiss" buttons. A status bar at the very bottom displays "Retrieving data - Succeeded".

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:*

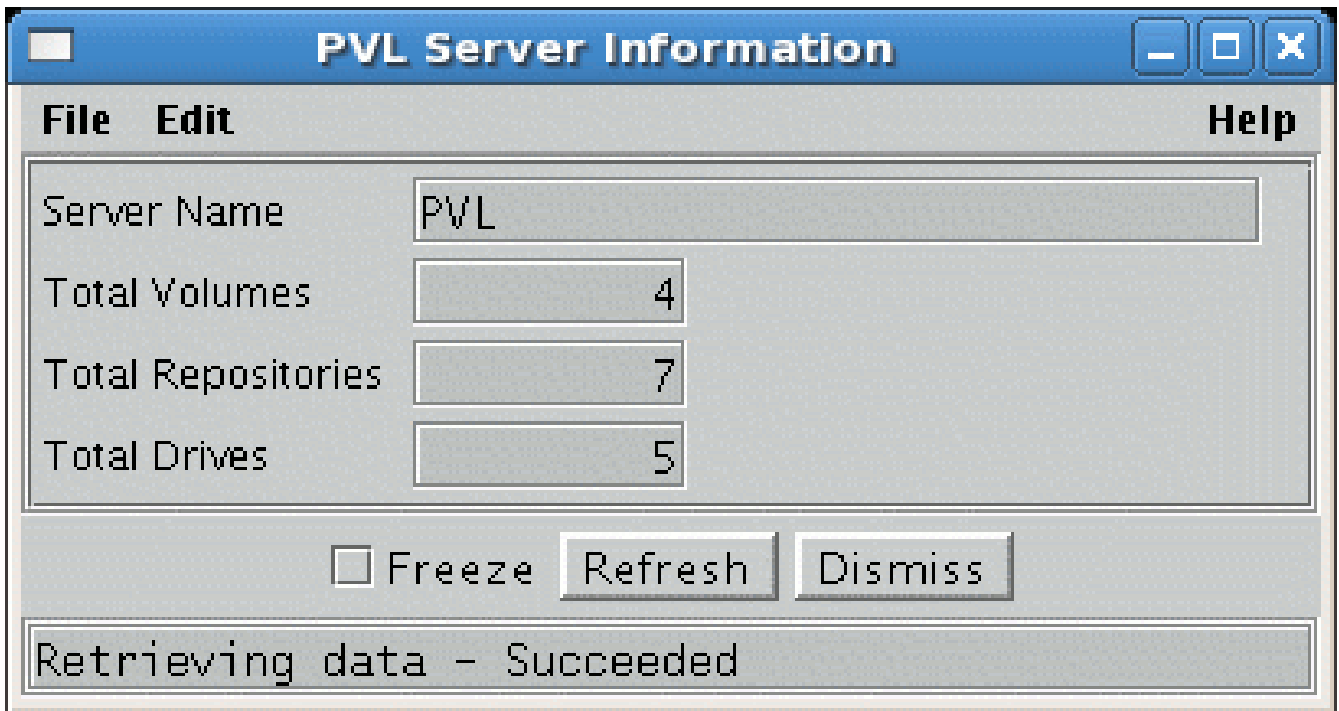
1. The Mover statistics (**Number of Requests Processed**, **Number of Request Errors**, **Number of Data Transfers**, **Number of Bytes Moved**, and **Total Move Time**) may all be reset to zero. If any of these fields are reset the **Time of Last Statistics Reset** field will also be updated with the 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.*

### **12.3.2.5. 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.

#### **12.3.2.6. Operator PVR information window**

Operator PVR Server Information

File Edit Help

Server Name: Operator PVR

Total Cartridges: 0

Cartridge Capacity: 1000

Cartridge Alarm Threshold: 90 percent

Shelf Tape Check-In Retry: 30 seconds

Shelf Tape Check-In Alarm: 10 minutes

Dismount Delay: 15 minutes

Defer Dismount Exempt Count: -1

Characteristics

Defer Dismounts  Support Shelf Tape

Freeze

Retrieving data - Succeeded

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

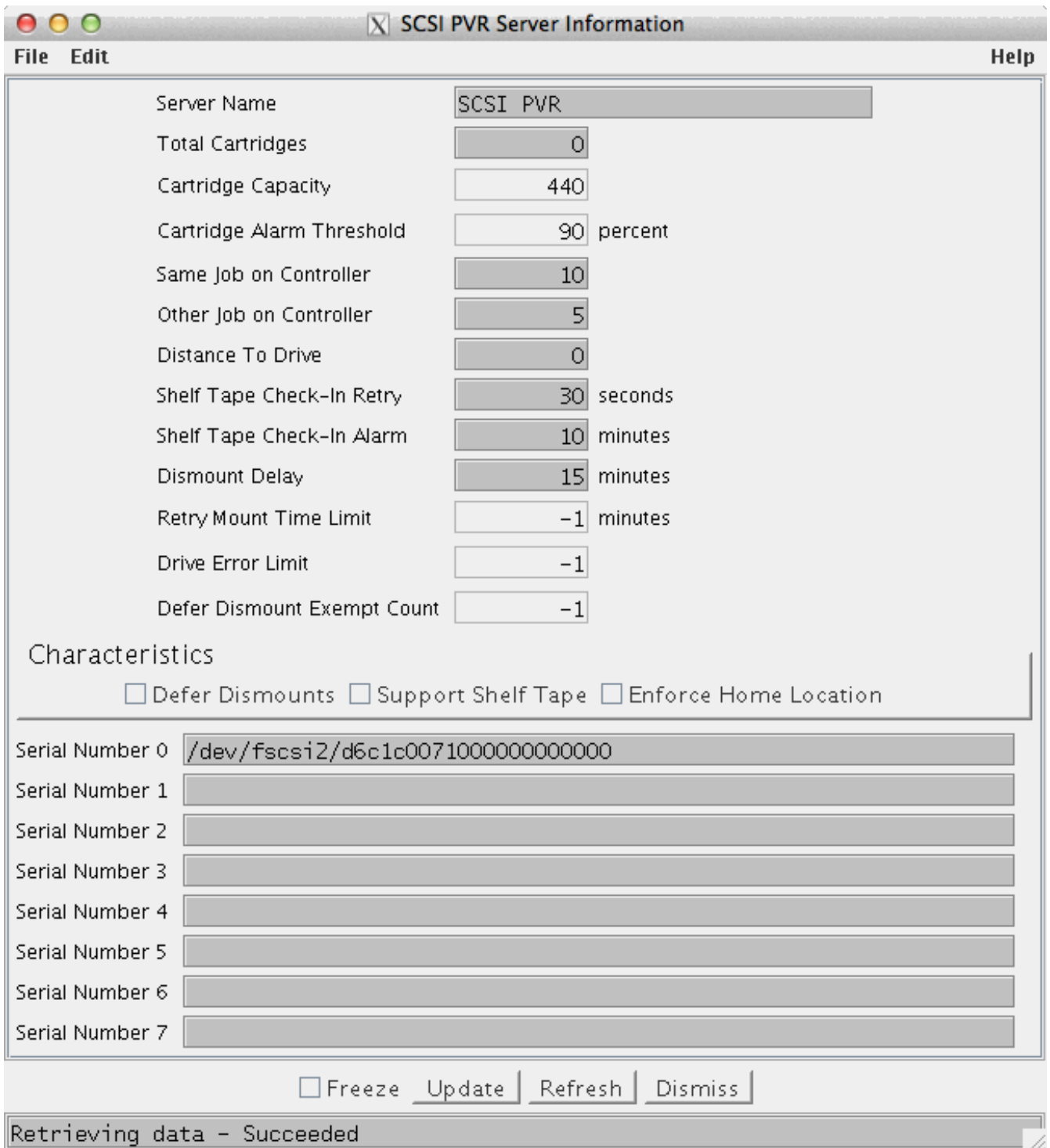
If ON, the PVL will delay the dismounting of a tape cartridge until the drive is required by another job or until the **Dismount Delay** time limit is exceeded. If the **Defer Dismount Exempt Count** is also set, the defer dismount logic will not be exercised on the number of drives set. This may result 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** utility.

### 12.3.2.7. 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:

$$\begin{aligned} \text{Score} = & \\ & \text{Weight 1} \times \text{Cartridges from this job mounted on this drive's controller} + \\ & \text{Weight 2} \times \text{Cartridges from other jobs mounted on this drive's controller} \\ & + \\ & \text{Weight 3} \times \text{Units of distance from the cartridge to the drive} \end{aligned}$$

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** utility.

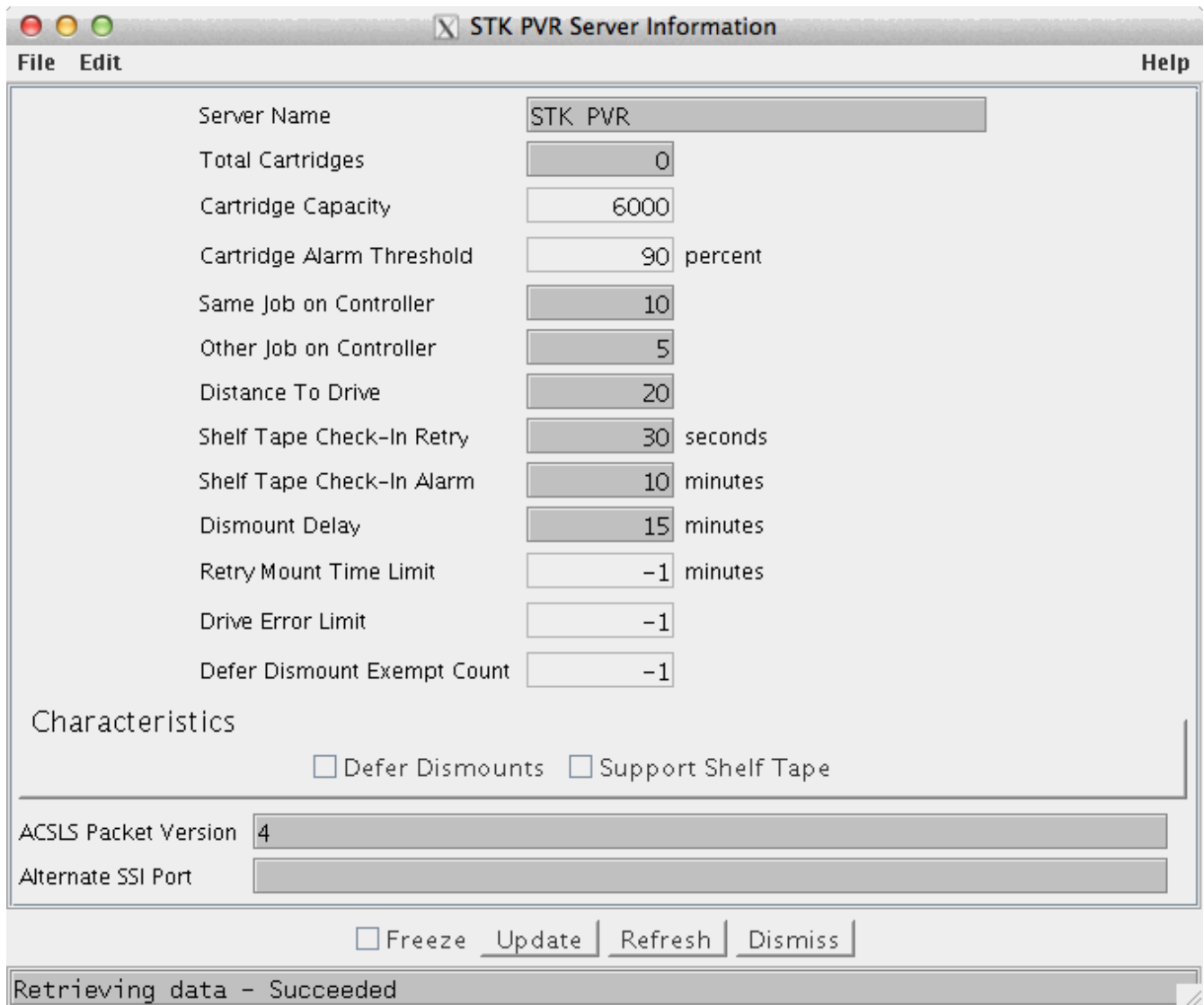
### Enforce Home Location

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` 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.

#### 12.3.2.8. 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:

$$\begin{aligned} \text{Score} = & \\ & \text{Weight 1} \times \text{Cartridges from this job mounted on this drive's controller} + \\ & \text{Weight 2} \times \text{Cartridges from other jobs mounted on this drive's controller} \\ & + \\ & \text{Weight 3} \times \text{Units of distance from the cartridge to the drive} \end{aligned}$$

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** utility.

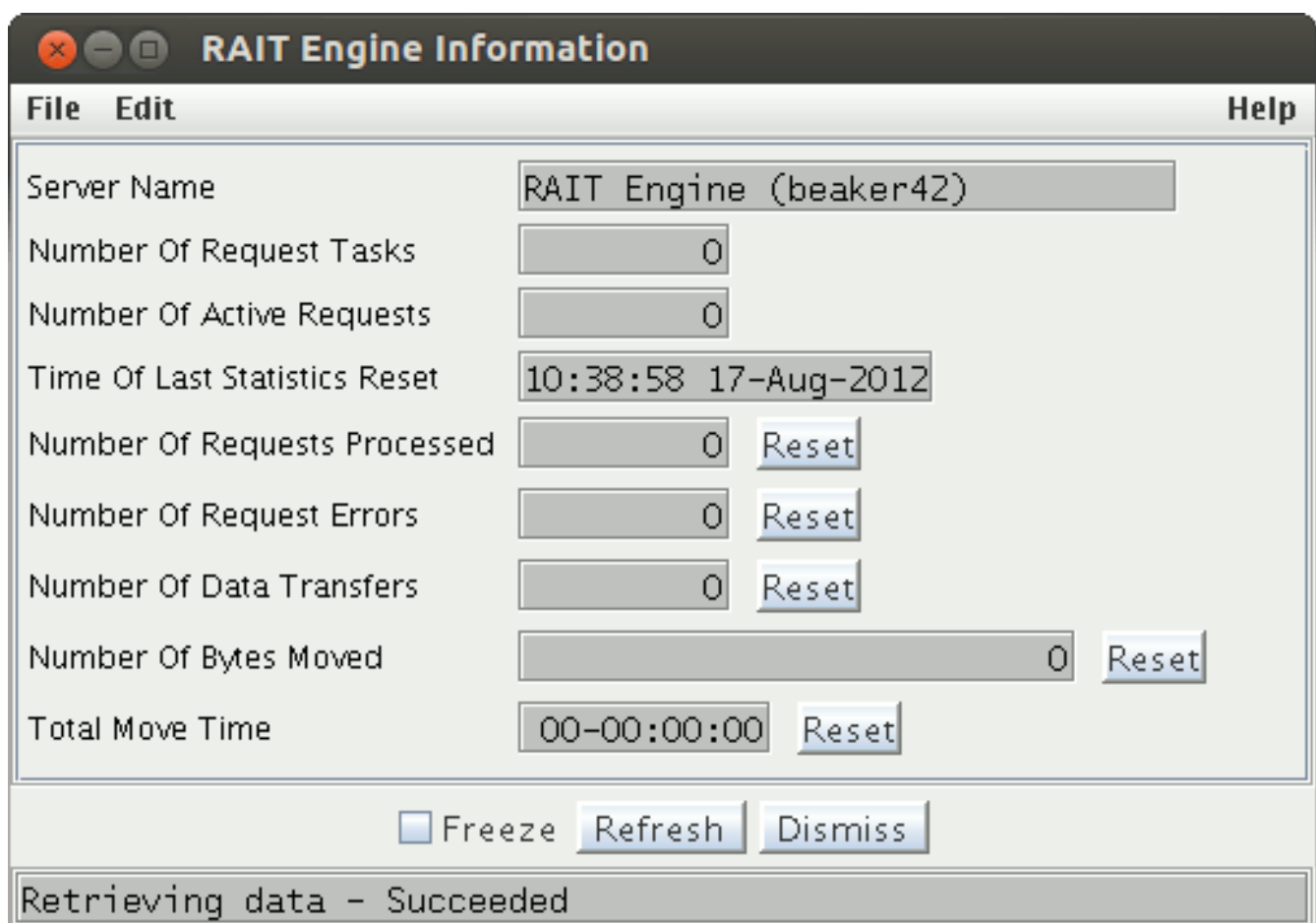
### ACSLs Packet Version

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](#).

#### 12.3.2.9. 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*

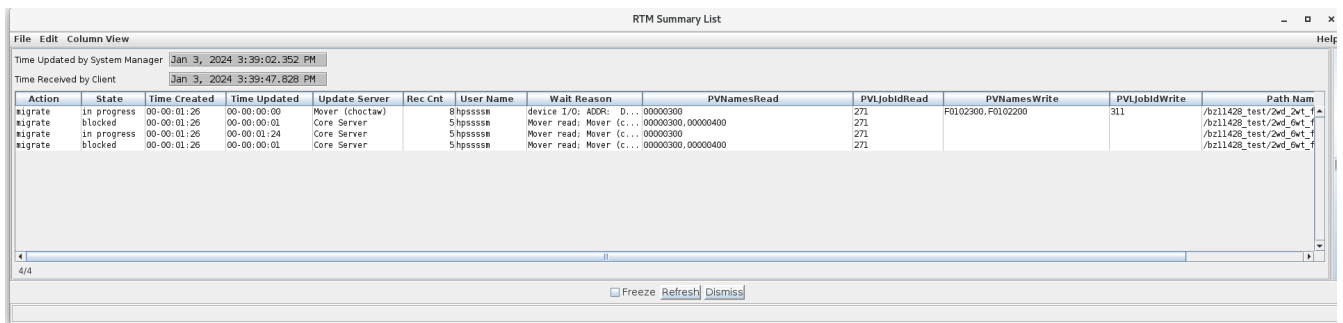
*HPSS Error Manual : Problem diagnosis and resolution* chapter : *Mover problems* section.

# 12.4. 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**) 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](#).

## 12.4.1. 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.

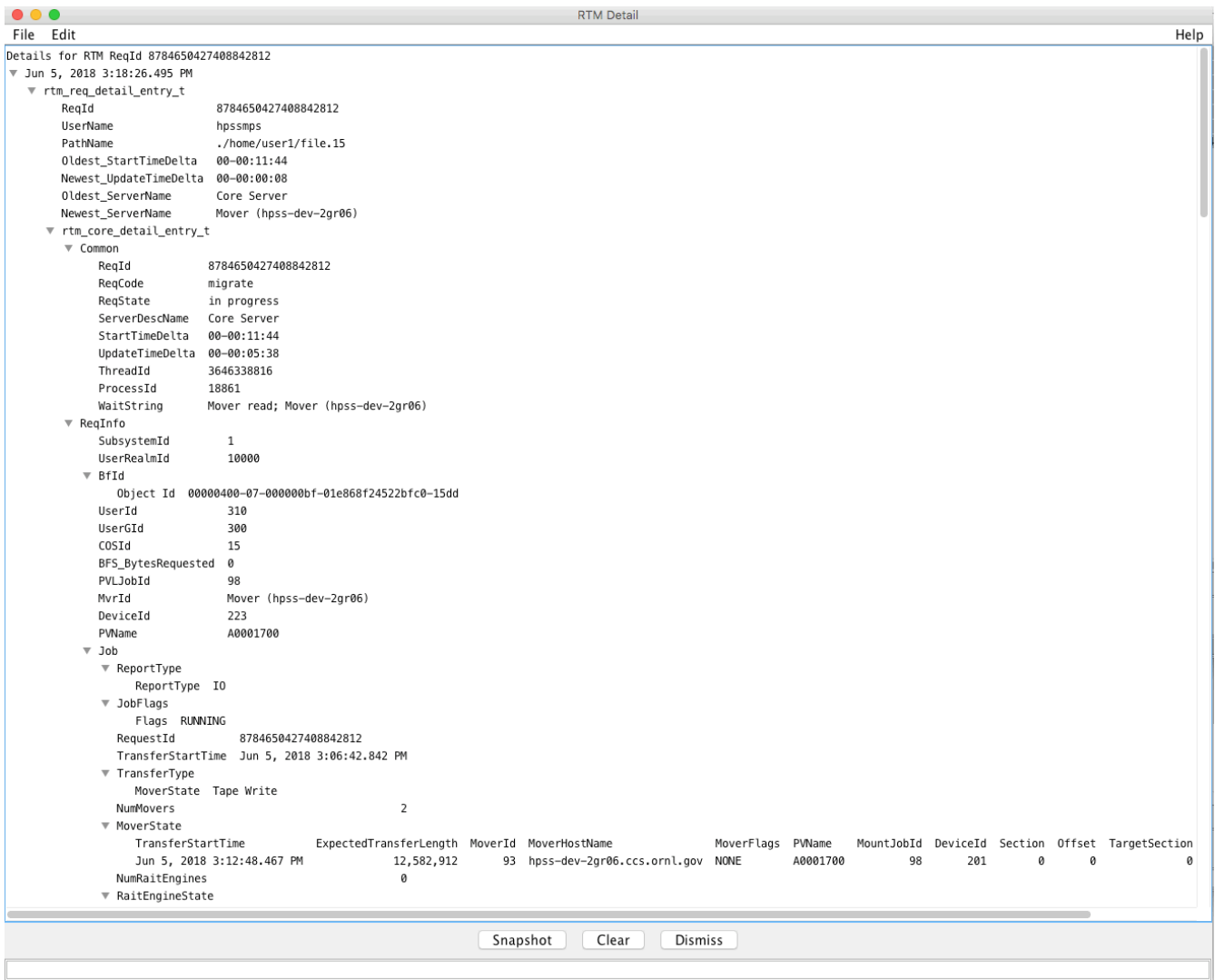
**PVLJobIdWrite**

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.

## 12.4.2. 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.

## 12.5. I/O Abort

This section describes how to abort I/O operations, and what to expect from HPSS following an abort request.

### 12.5.1. 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.

### 12.5.2. 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.

## 12.6. Starting HPSS

This section describes how to start and stop HPSS servers and prerequisite software.

### 12.6.1. Starting HPSS prerequisite software

Invoke the `/opt/hpss/bin/rc.hpss` script as root to start the HPSS prerequisite software such as DB2, Kerberos and LDAP. By default, stdout and stderr from commands started by `rc.hpss` 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` will cause stdout and stderr of commands invoked within the script to be sent to `/dev/console`, the terminal window where `rc.hpss` is running.

To start the prerequisite software, use the `-p` option to `rc.hpss`:

```
% su -  
% /opt/hpss/bin/rc.hpss -p [start]
```

### 12.6.2. 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.

#### 12.6.2.1. 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`:



```
% su -  
% /opt/hpss/bin/rc.hpss -d [start]
```

### 12.6.2.2. Starting SSM

The SSM System Manager configuration metadata should have already been created by **mkhps** 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**, 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** 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**:

```
% su -  
% /opt/hpss/bin/rc.hpss -m [start]
```

If the SSM's server name has changed since **mkhps** 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** script directly from the HPSS server machine if desired. However, most users of the SSM command line interface (**hpssadm**) will most likely invoke the script on the server machine.

### 12.6.2.3. 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.

#### 12.6.2.4. 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 22. 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](#).*

## 12.7. Stopping HPSS

### 12.7.1. 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` 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.

## 12.7.2. 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.

## 12.7.3. 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.

#### 12.7.4. 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**:

```
% su -  
% /opt/hpss/bin/rc.hpss -m stop
```

#### 12.7.5. Shutting down the Startup Daemon

To stop the Startup Daemon, use the **-d** option of **rc.hpss**:

```
% su -  
% /opt/hpss/bin/rc.hpss -d stop
```

#### 12.7.6. Stopping the prerequisite software

To stop the prerequisite software, use the **-p** option of **rc.hpss**:

```
% 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*.

## 12.8. Server repair and reinitialization

This section describes how to repair or reinitialize an HPSS server.

### 12.8.1. 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".

### 12.8.2. 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 23. Server reinitialization behavior*



Server	Behavior
Core Server	<p>Resets the <b>Maximum Active I/O Requests</b>, <b>Maximum Active Copy Requests</b>, <b>COS Copy to Disk</b>, <b>COS Change Retry Count</b>, <b>Tape Dismount Delay</b>, <b>Tape Handoff Delay</b>, <b>PVL Max Connection Wait</b> and <b>Fragment Smallest Block</b> values to the values in the specific configuration metadata record.</p> <p>Note that reducing the Maximum Active I/O Request 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.</p> <p>Reloads cached Class of Service information for the system.</p> <p>Reloads cached <b>Migration Policy</b> 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.</p> <p>Reloads cached global <b>Root User ID</b>, <b>Root is Superuser</b>, <b>Can change UID to self if has Control Perm</b>, <b>Can change UID if has Delete Perm on Security ACL</b>, <b>Object names can contain unprintable characters</b>, <b>Trashcans Enabled</b>, <b>Number of Trashcan Threads</b>, <b>Trash Incinerator Interval</b>, <b>Trash Unlink Eligible Time</b>, and <b>Trash Statistics Interval</b> values using values read from the global configuration metadata.</p> <p>Reloads cached global and subsystem <b>Metadata Space Warning Threshold</b>, <b>Metadata Space Critical Threshold</b>, <b>Metadata Space Monitor Interval</b>, <b>DB Log Monitor Interval</b>, and <b>COS Change Stream Count</b> values using values read from the global and subsystem configuration metadata.</p> <p>Reloads cached <b>Default Class of Service</b> and <b>Gatekeeper</b> values using values read from the subsystem configuration metadata.</p> <p>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.</p> <p>Reinitializes the in-memory cache of environment variable values - rereads the <code>env.conf</code> file.</p> <p>Adjusts the maximum number of open files supported by the Core Server when the number of migration streams</p>

Server	Behavior
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 <b>Storage Class Update Interval</b> , <b>Core Server API Failures Allowed</b> , and <b>MPS Report File Name</b> type-specific configuration values.
GK	Reloads the <i>Accounting Policy Options</i> flags.
All other servers	Update server basic configuration thread, request queue, and connection configuration.

example, security-related configuration settings.

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.



## 12.9. 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 13. Storage configuration

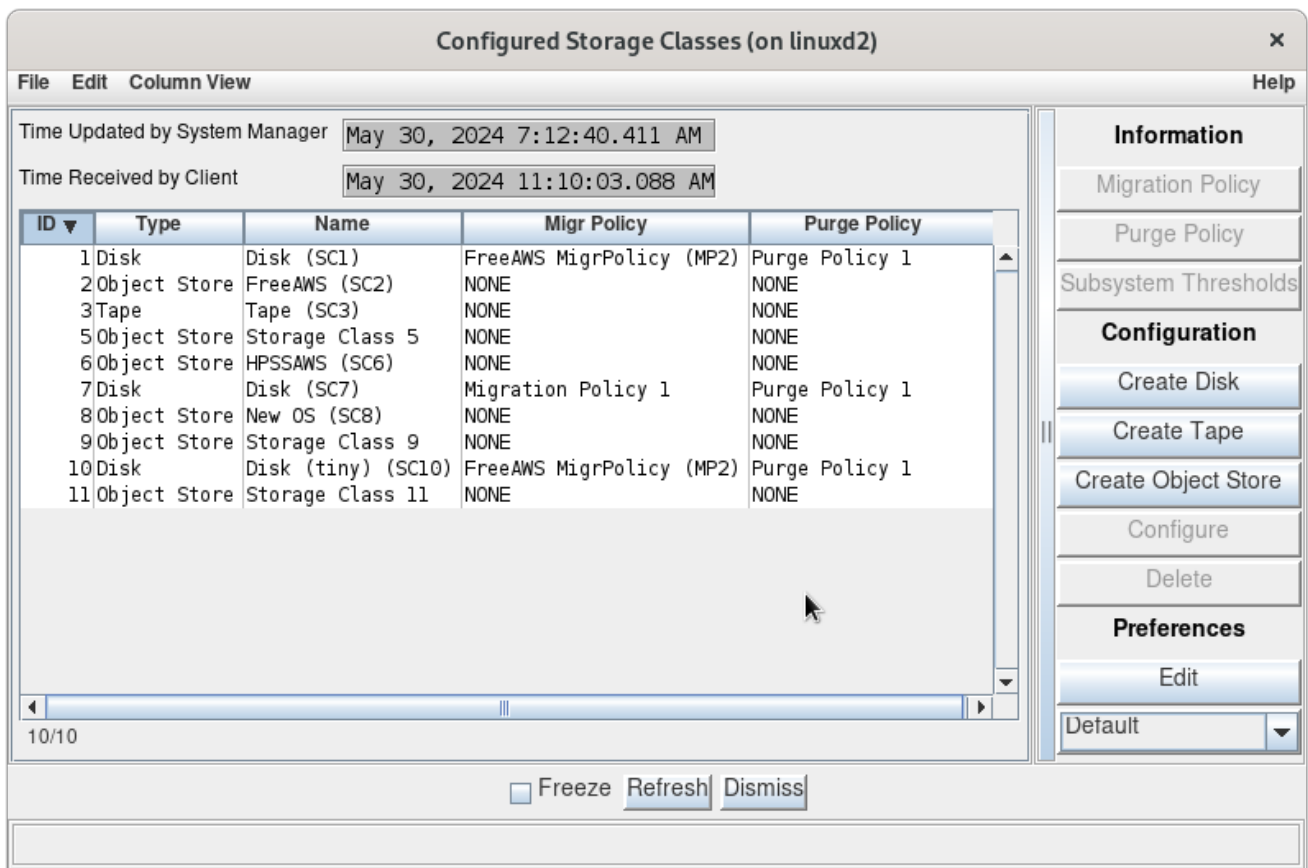
This chapter describes the procedures for creating, modifying, and deleting storage classes, hierarchies, Classes of Service, migration policies, purge policies, and file families.

## 13.1. Storage classes

This section describes the process of configuring storage classes.

### 13.1.1. Configured Storage Classes window

A storage class can be created and managed using the *Configured Storage Classes* window.



ID	Type	Name	Migr Policy	Purge Policy
1	Disk	Disk (SC1)	FreeAWS MigrPolicy (MP2)	Purge Policy 1
2	Object Store	FreeAWS (SC2)	NONE	NONE
3	Tape	Tape (SC3)	NONE	NONE
5	Object Store	Storage Class 5	NONE	NONE
6	Object Store	HPSSAWS (SC6)	NONE	NONE
7	Disk	Disk (SC7)	Migration Policy 1	Purge Policy 1
8	Object Store	New OS (SC8)	NONE	NONE
9	Object Store	Storage Class 9	NONE	NONE
10	Disk	Disk (tiny) (SC10)	FreeAWS MigrPolicy (MP2)	Purge Policy 1
11	Object Store	Storage Class 11	NONE	NONE

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.

### 13.1.2. 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 24. 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 25. 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 26. 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

### 13.1.3. Disk Storage Class Configuration

**Disk Storage Class Configuration**

File Edit Help

Storage Class ID: 5

Storage Class Name: Disk - Storage Class 5

Storage Class Type: Disk

Migration Policy: 5 | Migration Policy 5

Purge Policy: 3 | Purge Policy 3

Warning Threshold: 80 percent

Critical Threshold: 90 percent

Optimum Access Size: 0

Average Number of Storage Segments: 4

**Storage Segment Size**

Media Type: Generic - Default Disk

Media Block Size (MBS): 4KB

VV Block Size (VVBS): 1MB

Data Stripe Width (DSW): 1

Data Stripe Length (DSL): 1MB (VVBS \* DSW)

PV Size (PVSIZE): 512MB

VV Size (VVSIZE): 512MB (PVSIZE \* DSW)

Min Multiplier (MINMULT): 1

Min Storage Segment Size (MINSEG): 1MB (MINMULT \* DSL)

Max Multiplier (MAXMULT): 1 (2<sup>0</sup>)

Max Storage Segment Size (MAXSEG): 1MB (MAXMULT \* MINSEG)

Min Segments in VV (MINSVV): 512 (VVSIZE / MAXSEG)

Max Segments in VV (MAXSVV): 512 (VVSIZE / MINSEG)

**Transfer Rate**

Device I/O Rate (DIOR): 3072 KB/sec

Stripe Transfer Rate (STR): 3072 KB/sec (DSW \* DIOR)

Update Delete Start Over Clone (partial) Clone (full) Dismiss

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** utility to correct this problem. See the **mkmprec** man page for more information.*

## 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 (VVSIZ)** 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 (VVSIZ)** 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)**.

## **13.1.4. Tape Storage Class Configuration**

Tape Storage Class Configuration

File Edit Help

---

Storage Class ID

Storage Class Name

Storage Class Type **Tape**

Migration Policy

Warning Threshold  volumes

Critical Threshold  volumes


Optimum Access Size

Average Latency  seconds


Maximum VVs To Write

---

**Storage Segment Size**

Media Type  


Block Protection


Media Block Size (MBS)  

VV Block Size (VVBS)

Data Stripe Width (DSW)

Data Stripe Length  (VVBS \* DSW)


PV Size (PVSIZE)  


VV Size (VVSIZE)  (PVSIZE \* DSW) 

Max Storage Segment Size


---

**Transfer Rate**


Device I/O Rate (DIOR)  KB/sec 

Stripe Transfer Rate (STR)  KB/sec (DSW \* DIOR) 

Seconds Between Tape Marks (SBTM)

Blocks Between Tape Marks (BBTM)  ((SBTM \* DIOR \* 1024) / MBS) 

Seconds Between Unbuffered Tape Marks (SBUTM)

Blocks Between Unbuffered Tape Marks (BBUTM)  ((SBUTM \* DIOR \* 1024) / MBS) 

---

**RAIT Options**

Parity Stripe Width  (0 = RAIT is disabled)

Minimum Write Parity

Number of RAIT Engines

Read Verification       Mount Minimum PVs

---

---

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** utility to correct this problem. See the **mkmprec** man page for more information.*

#### 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 (PVSIZ)

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 (VVSIZ)

The virtual volume size. The product of the **PV Size (PVSIZ)** 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 (VVSIZ)**).

### 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).

## **13.1.5. Object Store Storage Class Configuration**



# Object Storage Endpoint Storage Class Configuration (on linuxd2)



File Edit

Help

Storage Class ID

Storage Class Name

Storage Class Type **Object Store**

Warning Threshold  percent

Critical Threshold  percent

Optimum Access Size

Average Latency  seconds

## General Settings

Media Type

Region

Object Store Storage Class

Stage Allowed       SOID Object Names

## Restore Settings

Restore Priority

Restore Duration Days

Restore Notifications

Restore Notification Port

## Multipart Settings

Part Size MiB

Number of Parts

Number of Concurrent Parts (NCP)

## Transfer Rate

Device I/O Rate (DIOR)  KB/sec

Multipart Transfer Rate (MTR)  KB/sec (NCP \* DIOR)

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 flag 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 enable 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).*

## 13.1.6. 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.

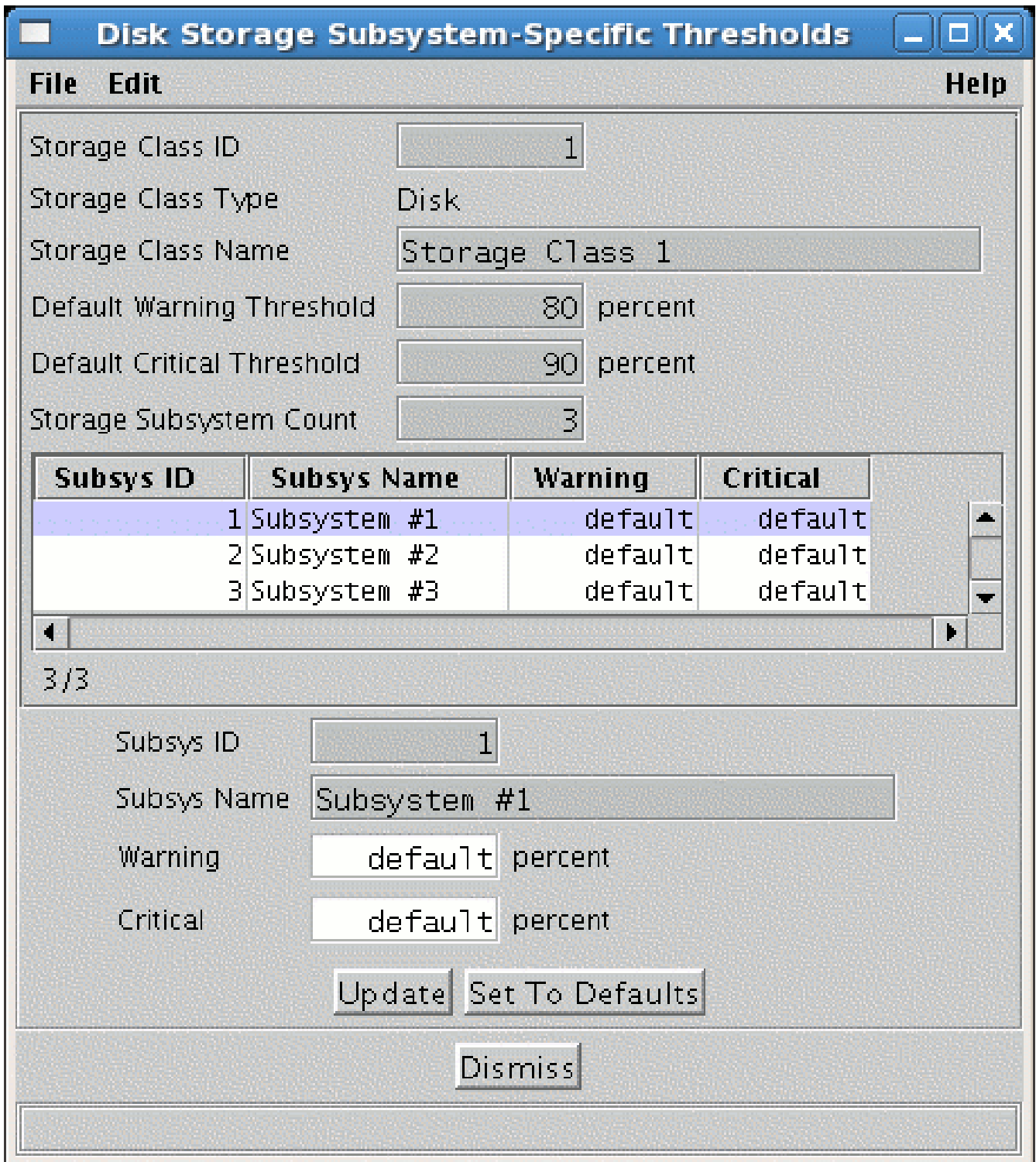
Each storage class configuration window has fields for **Warning Threshold** and **Critical Threshold**. By default, these thresholds apply to all storage subsystems using the storage class. However, the storage class thresholds can be overridden by configuring a subsystem-specific threshold allowing 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](#)

### 13.1.6.1. 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](#)

### 13.1.6.2. Tape Storage Subsystem-Specific Thresholds

Storage Class ID: 32

Storage Class Type: Tape

Storage Class Name: Storage Class 11

Default Warning Threshold: 10 volumes

Default Critical Threshold: 5 volumes

Storage Subsystem Count: 3

Subsys ID	Subsys Name	Warning	Critical
1	Subsystem #1	default	default
2	Subsystem #2	default	default
3	Subsystem #3	default	default

3/3

Subsys ID: 1

Subsys Name: Subsystem #1

Warning: default volumes

Critical: default volumes

Update Set To Defaults

Dismiss

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*

### 13.1.7. 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** utility to create migration records for these files so that they will migrate properly. See the **mkmprec** man page for more information.

#### 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.

### 13.1.8. 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** utility, which can quickly verify the number of files in each COS.

It is recommended you call HPSS support for assistance with this operation.

## 13.2. Storage hierarchies

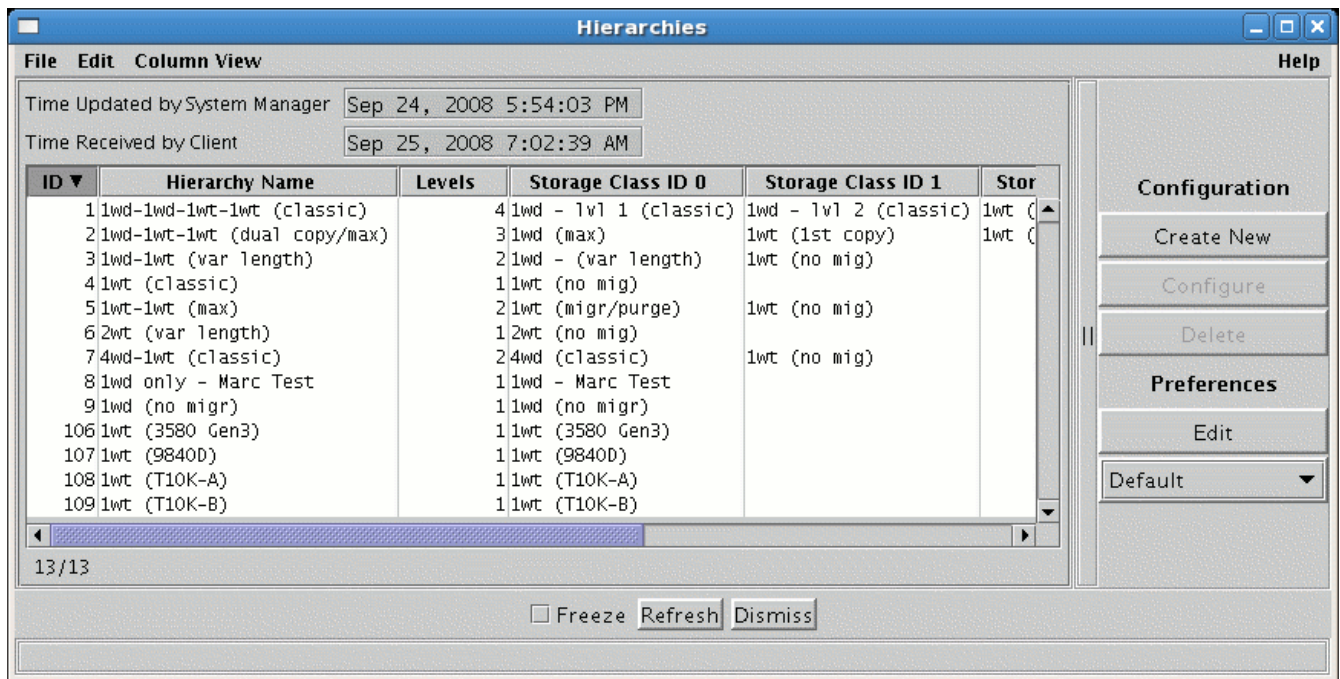
This section describes how to create, modify, and delete storage hierarchies.

### 13.2.1. 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.

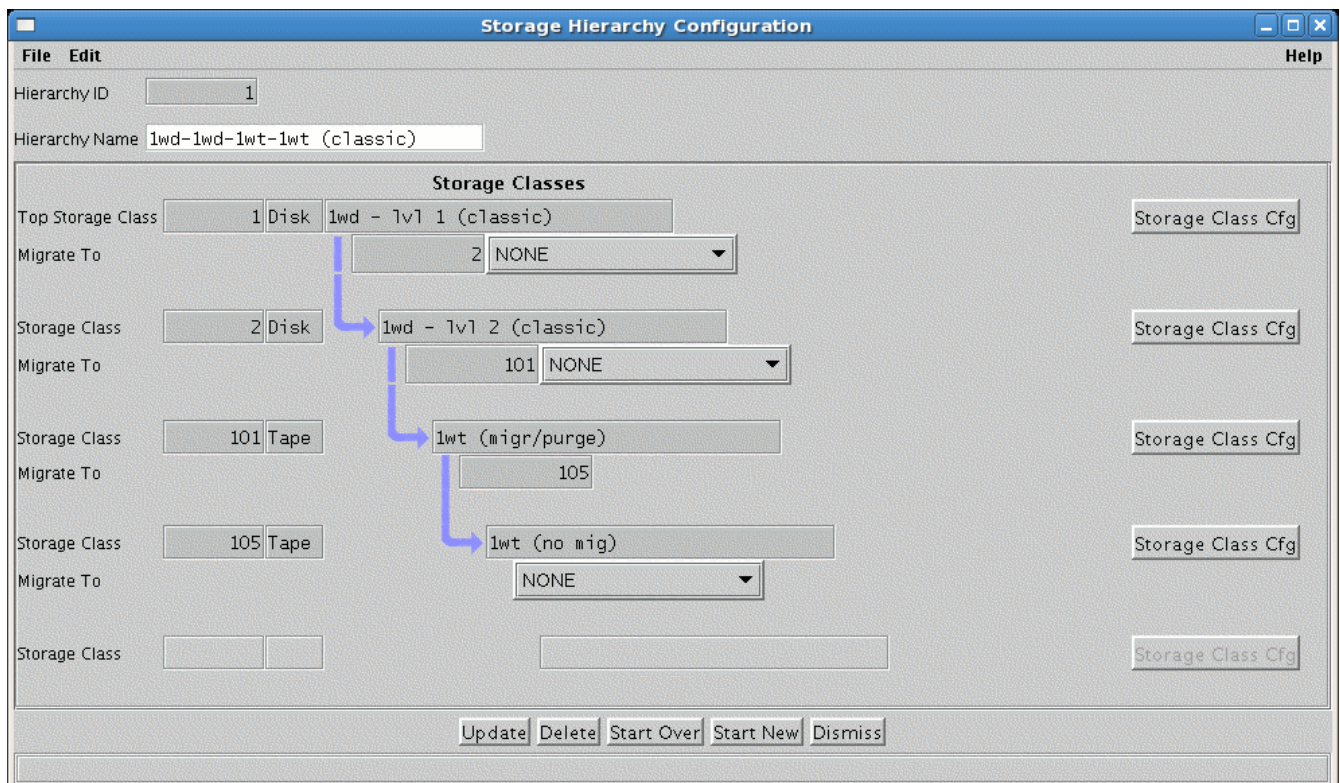
#### Configure

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.

### 13.2.2. 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.

### 13.2.3. 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** utility.

### 13.2.4. 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.

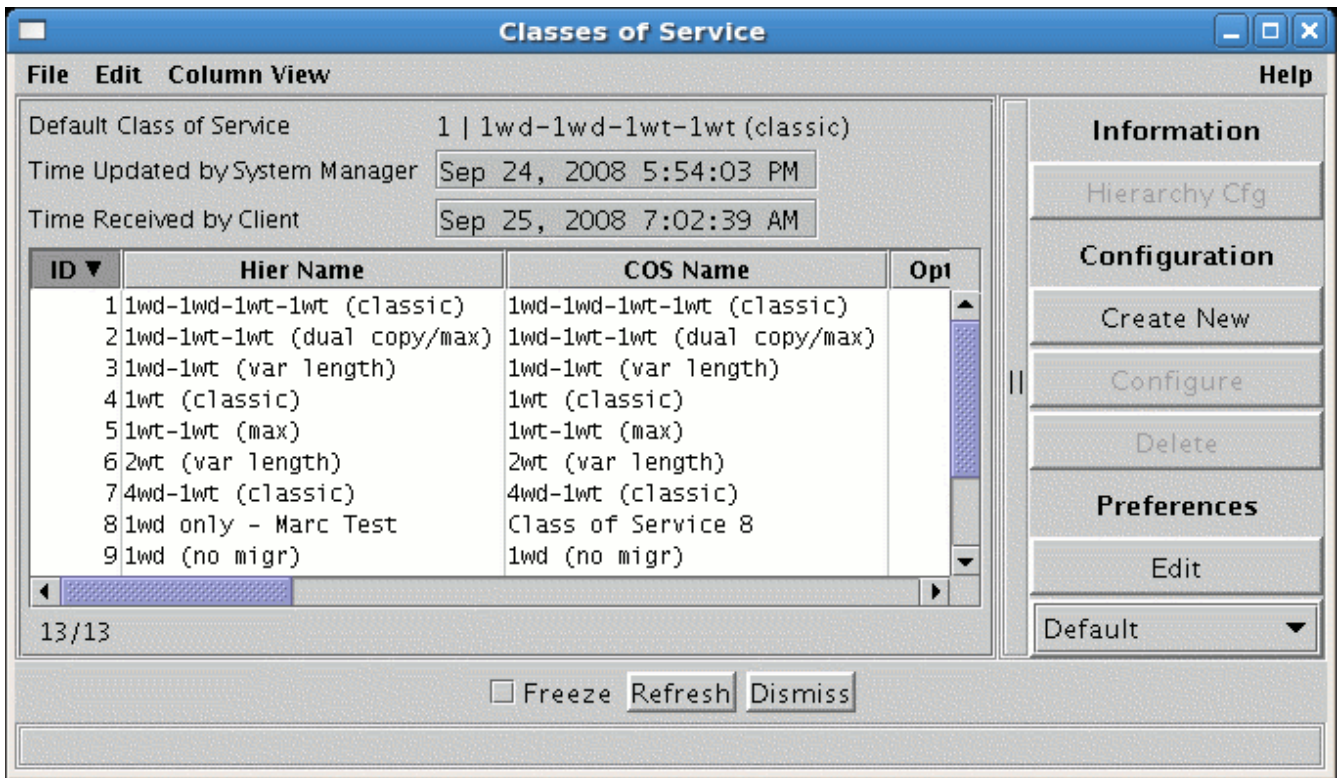
It is recommended you call HPSS support to assist with this operation.

## 13.3. Classes of Service

This section describes how to configure Classes of Service.

### 13.3.1. 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.

### **13.3.2. Class of Service Configuration window**

**Class of Service Configuration (on hogwarts6)** \_ □ ×

**File Edit** **Help**

---

Class ID

Class Name

Storage Hierarchy  ▼

Stage Code  ▼

Minimum File Size  bytes

Maximum File Size  bytes

Allocation Method  ▼

Hash Algorithm  ▼

Migration Order  ▼

**Class Flags**

Enforce Maximum File Size     Force Selection

Auto Stage Retry                     Auto Read Retry

Auto Change COS                         Truncate Final Segment

Full Aggregate Recall

**Class Characteristics**

Access Frequency  ▼

Optimum Access Size  bytes

Average Latency  seconds

Transfer Rate  KB/sec

**R/W Operations**

Read     Write     Append

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 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 ordering is 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 `hpsssm` 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.*

### 13.3.3. 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.

#### 13.3.4. 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** 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.

#### 13.3.5. 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 **scrub** utility can be used to initiate the COS change request of a file as follows:

```
% scrub
scrub> changecos <fullPathname> <newCOSid>
scrub> quit
```

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.

### 13.3.6. 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 scrub utility can be used to cancel the COS change request as follows:

```
% scrub
scrub> changecos <fullPathname> 0
scrub> quit
```

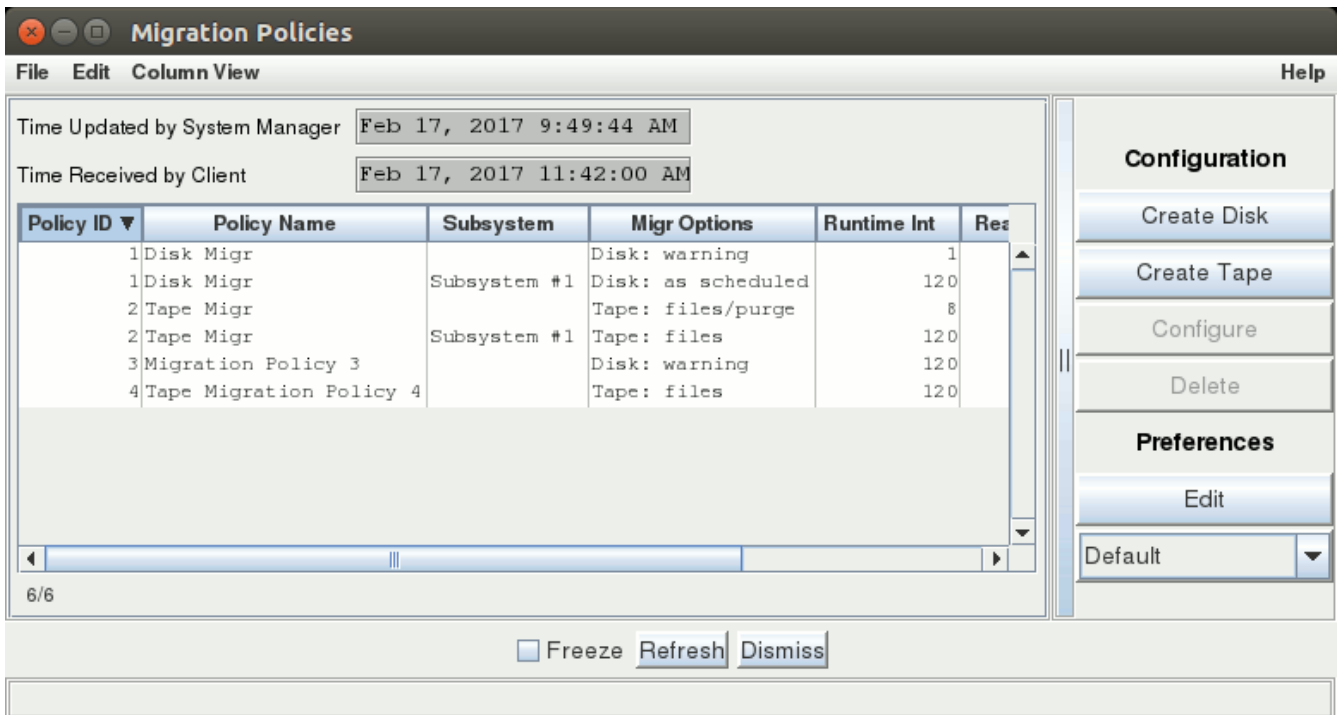
The associated Core Server COS change record will be removed immediately. However, this request may fail if the existing COS change request is currently being processed by one of the Core Server's COS change threads. In this case, the user should attempt to reissue the request later.

## 13.4. 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.

### 13.4.1. 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.

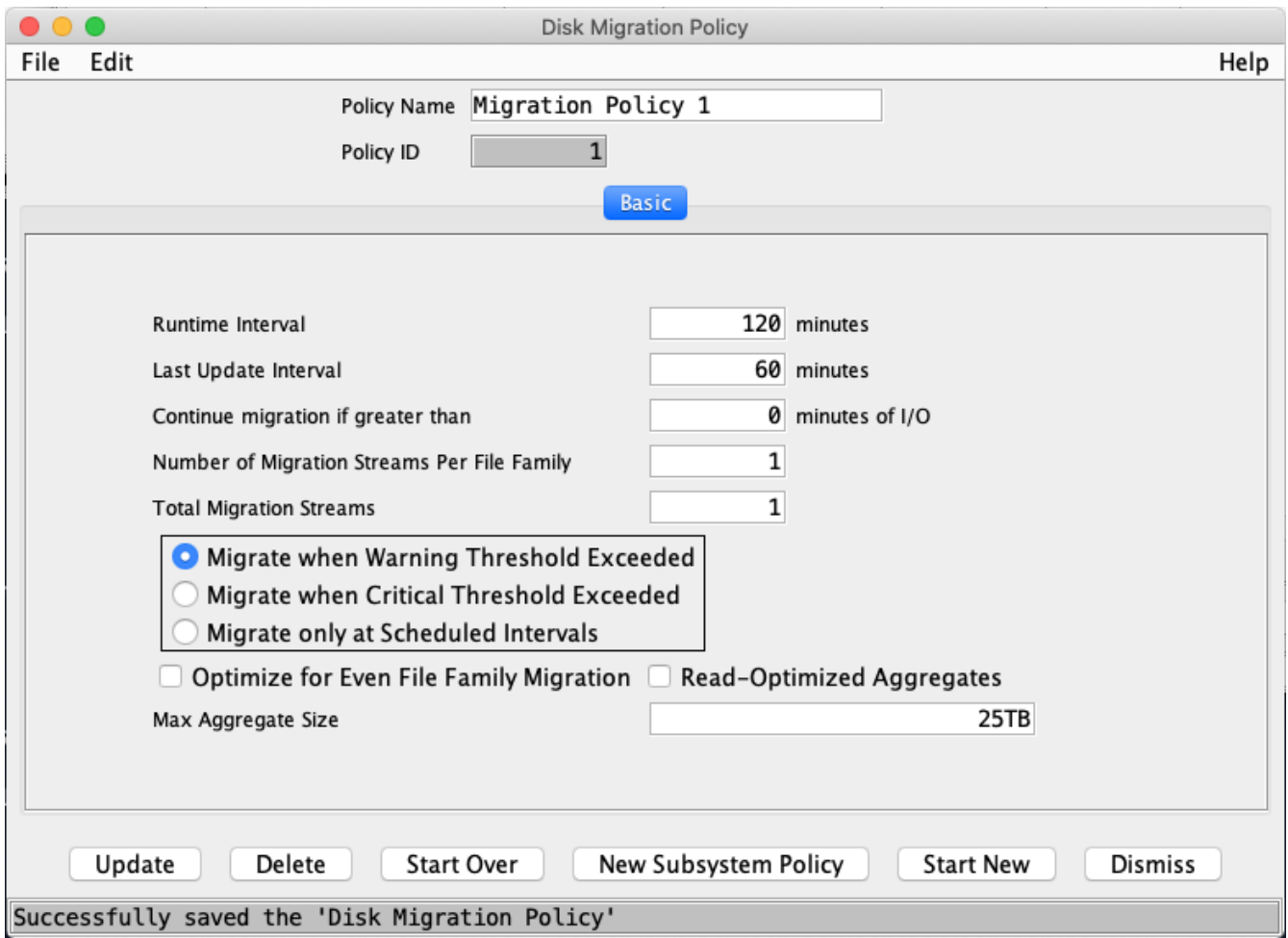
### *Related information*

Section [Migration/Purge Server](#)

## **13.4.2. Migration policy configuration**

This section describes the configuration, update, and deletion of migration policies.

### **13.4.2.1. 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. This value effectively determines how many file families can be migrated simultaneously. 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).

### **Total Migration Streams**

The number of parallel migration threads which 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*.

## 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:

$$\begin{aligned} & (SW \times TMS)_{Hier1,LevelA} + (SW \times TMS)_{Hier1,LevelB} + \dots + \\ & (SW \times TMS)_{Hier2,LevelA} + (SW \times TMS)_{Hier2,LevelB} + \dots + \\ & \dots + \\ & (SW \times TMS)_{Hiern,Levelx} + (SW \times TMS)_{Hiern,Levely} + \dots + \end{aligned}$$

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 Aggregated** 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

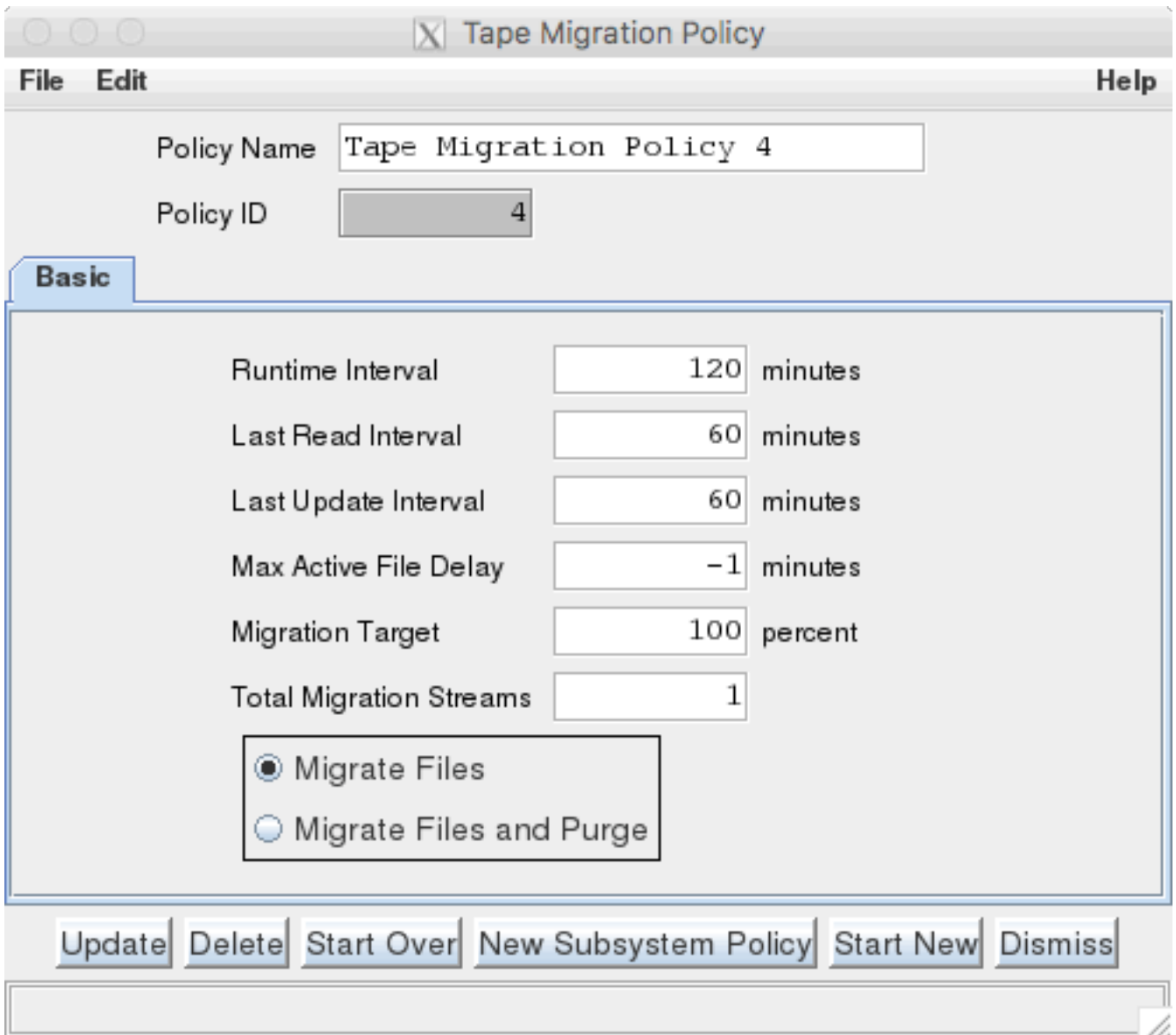
The maximum total size in bytes of a tape aggregate. Disk-to-tape migration bundles bitfiles into tape aggregates, and this value sets the maximum aggregate size. Tape aggregates are limited to a maximum of the total estimated size of the destination tape volume. Tape aggregation can be disabled by setting the "Max Aggregate Size" to 0.



### Advice

\_For details about migration of files to aggregates on tape, see [Overview of tape aggregation](#).

## 13.4.2.2. 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](#)

#### **13.4.2.3. 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.

#### **13.4.2.4. 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.

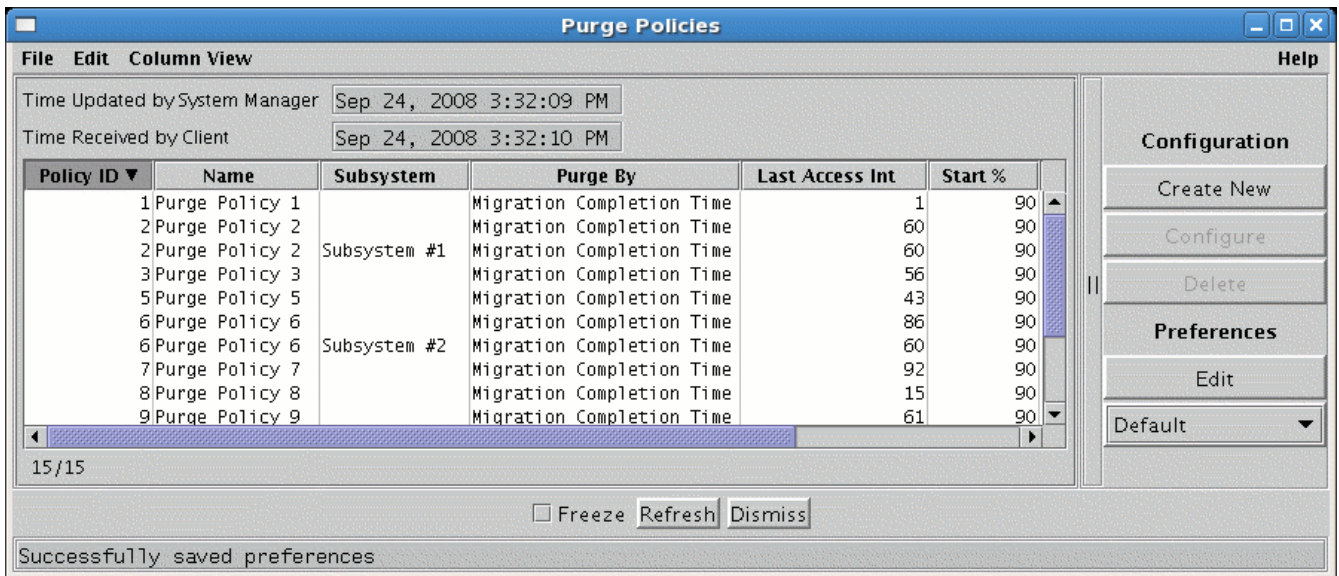
## 13.5. Purge policies

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.

### 13.5.1. 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.

## 13.5.2. Purge Policy configuration

**Purge Policy**

File Edit Help

Policy Name

Policy ID

Basic Subsystem 1

Do not purge files last accessed within  minutes

Start purge when space used exceeds  percent

Stop purge when space used falls to  percent

Purge locks expire after  minutes

Purge By

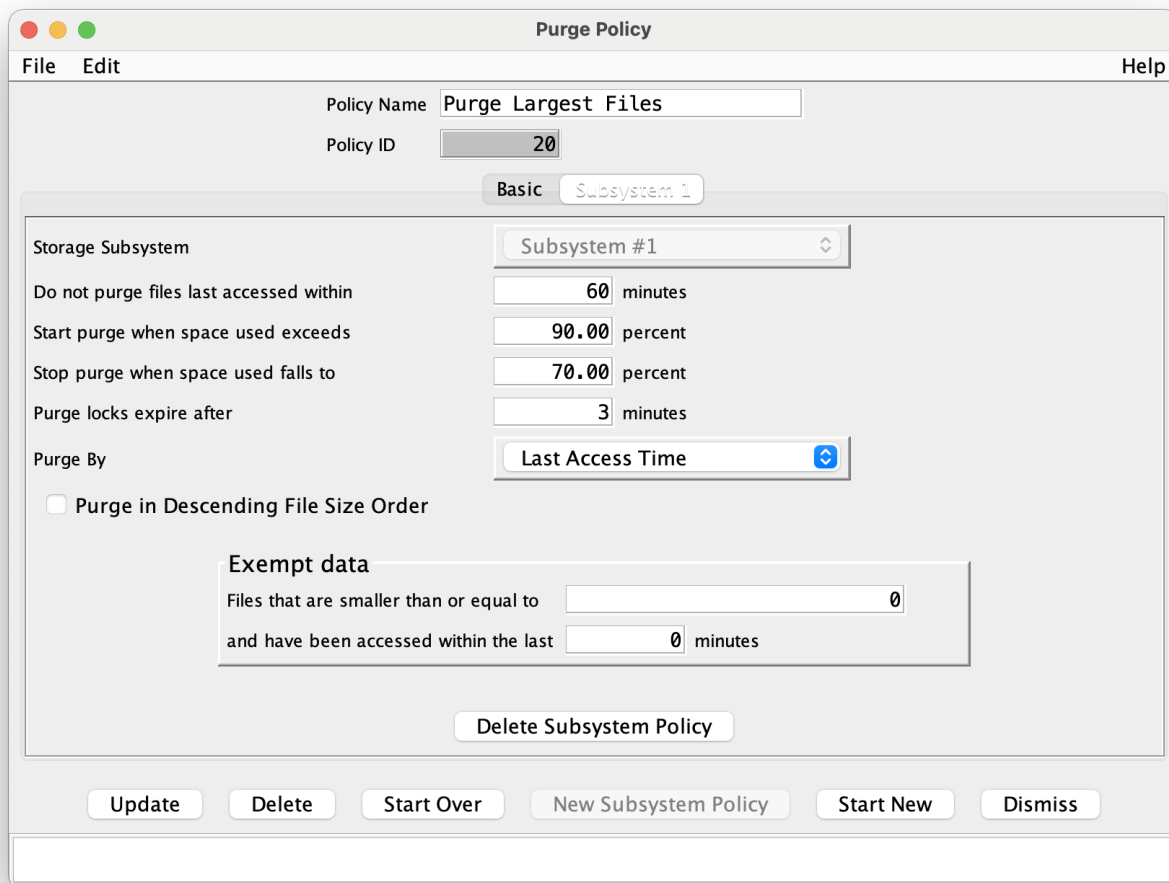
Purge in Descending File Size Order

**Exempt data**

Files that are smaller than or equal to

and have been accessed within the last  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**) 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** utility. Super purge locks should be rare. They are set temporarily by the **recover** utility and released when the **recover** 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](#)

## **13.5.3. 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.

### 13.5.4. 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.

### 13.5.5. Purge parallelism

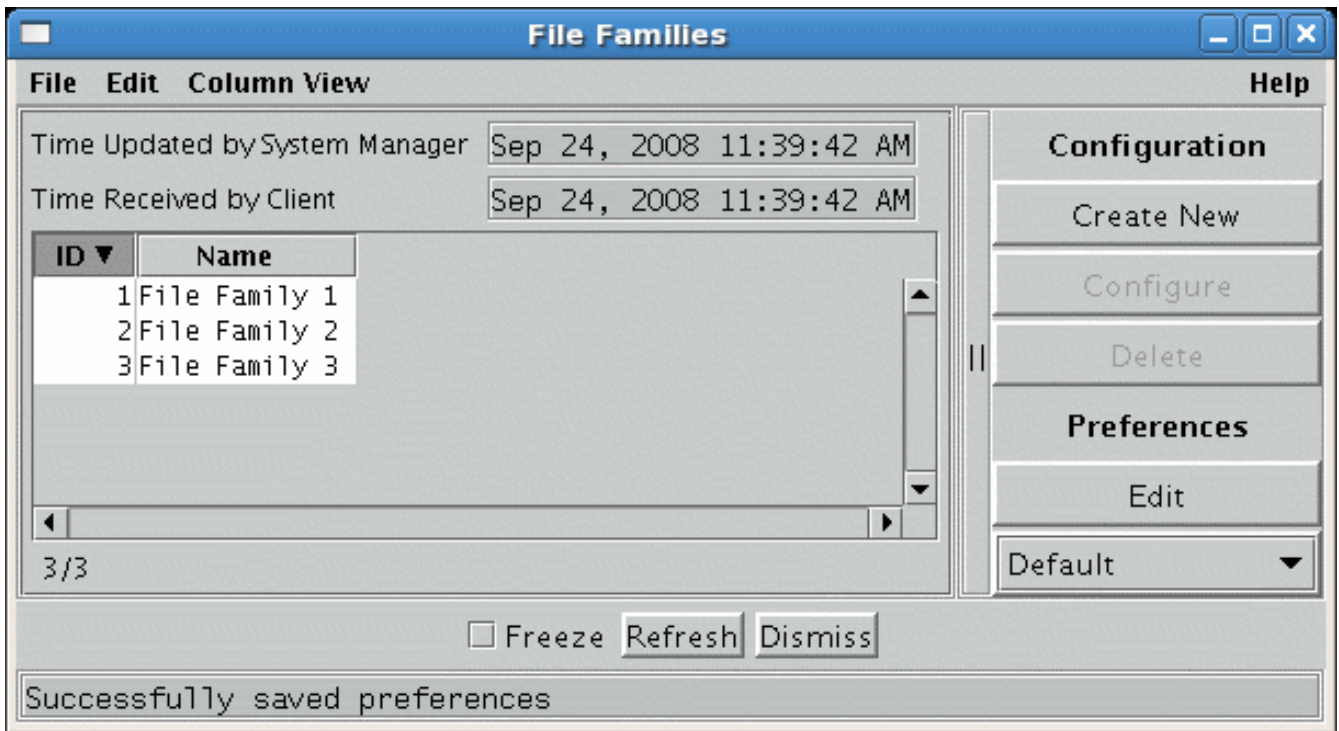
The degree of parallelism used for purging files from disk storage classes can be controlled via the HPSS environment. By default, the MPS degree of parallelism is set to two threads per storage class. You *may* find that increasing the number of purge threads increases the speed at which your HPSS system 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.

## 13.6. 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.

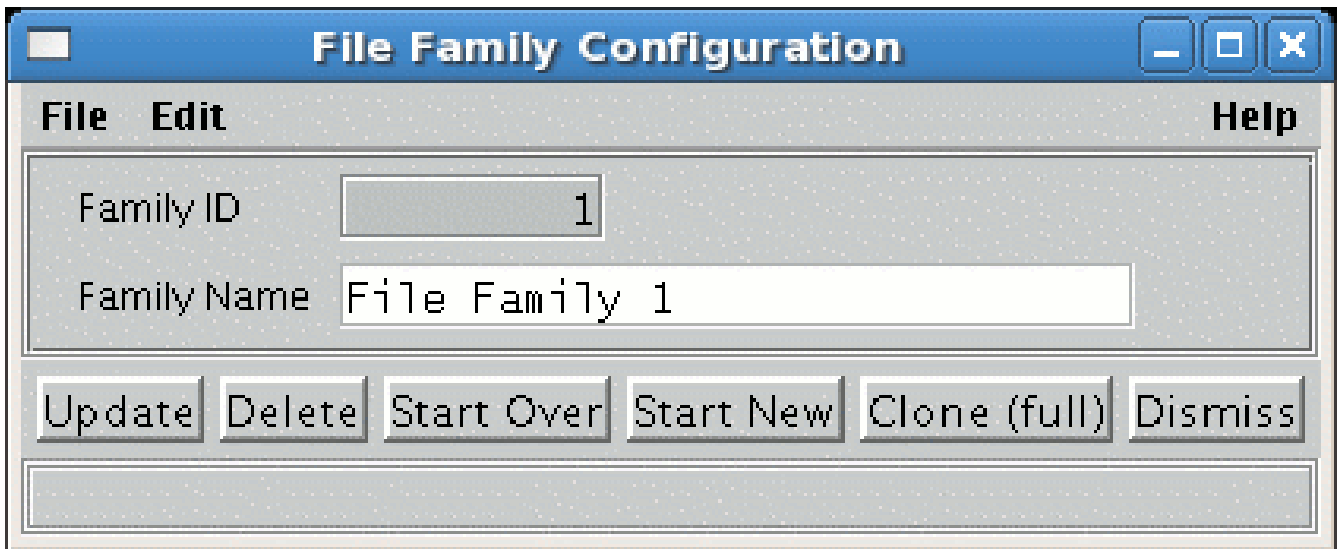
##### **Configure**

Open the selected file family configurations for editing.

##### **Delete**

Delete the selected file family configurations.

### **13.6.1. 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>".

### **13.6.2. 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.

### **13.6.3. 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 14. 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.

## 14.1. 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 (see [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 [Devices and Drives window](#).

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** 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

Disk Device Configuration (on nudge.hpslab.ibm.com)
✕

File Edit
Help

Device ID

Device Type Generic - Default Disk ▼

Mover Mover (nudge) ▼

**Basic Information**

Device Name

Media Block Size

Capacity

Starting Offset

SAN ID

**Device Flags**

Read Enabled  
  Write Enabled  
  SAN3P-transfer Enabled  
  Multiple Mover Tasks  
 Disable Direct I/O  
  Enable LibAIO

Drive Address

Controller ID

Maintenance Date  (ex. Dec 31, 2024 10:59:30.501 AM)

Mounted Volume

Comment

Update
Delete
Start Over
Clone (partial)
Clone (full)
Dismiss

## Tape Device Configuration



**Tape Device Configuration** \_ □ ×

**File Edit** **Help**

Device ID


Device Type

Mover

**Basic Information**

Device Name

**Device Flags**

Read Enabled   
  NO-DELAY Support   
  Removable Media Support  
 Write Enabled   
  TAOS Support   
  Locate Support   
 Fast Positioning   
  Disable RAO

PVR

Drive Address

Controller ID

Polling Interval  seconds (-1 to disable)

Maintenance Date  (ex. Dec 31, 2021 10:59:30.672 AM)

Comment

Library Unit ID


Update the selected device/drive: Drive must be locked.


## Object Store Device Configuration

**Object Storage Endpoint Device Configuration (on linuxd2)**


**File Edit Help**

Device ID

Device Type **AWS - S3 Object Store** 

Mover **Mover (linuxg1)** 

**Basic Information**

Device Name  

**Device Flags**

Read Enabled  Write Enabled  Multiple Mover Tasks

Controller ID

Mounted Volume

Comment

**Add Start Over Clone (partial) Clone (full) Dismiss**

Setting field: Device Type

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` utility with the `-i` option and device name. The UUID for a SAN disk device can be determined by running the `hpss_san3p_part` 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*

[What is TAOS?](#)

*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` 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

<b>Device driver</b>	<b>Mover executable</b>	<b>NO-DELAY support</b>	<b>Locate support</b>	<b>TAOS</b>	<b>Fast Positioning</b>	<b>Disable RAO</b>
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** 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 indication of the adapter/bus that is in the data path to the device.



### *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.*

## 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.

## 14.1.1. Devices and Drives window

The screenshot shows a window titled "Devices and Drives (on linux2)". It features a menu bar with "File", "Edit", "Column View", and "Help". Below the menu bar, there are two text boxes: "Time Updated by System Manager" with the value "Feb 5, 2024 2:03:55.378 PM" and "Time Received by Client" with the value "Feb 5, 2024 2:03:55.380 PM".

ID	Device Type	Device Name	Device State	Drive State	De
1	Generic - Default Disk	/hpss_disk1	Enabled	Enabled	Unlc
2	AWS - S3 Object Store	https://s3.amazonaws.com	Enabled	Enabled	Unlc
4	IBM - 3592 E07/EH7 (TS1140) Tape	/dev/st1	Enabled	Enabled	Unlc
5	IBM - 3592 E07/EH7 (TS1140) Tape	/dev/st0	Enabled	Enabled	Unlc
6	AWS - S3 Object Store	https://s3.amazonaws.com	Enabled	Enabled	Unlc
7	AWS - S3 Object Store	http://s3.tidus.clearlake.ibm.com:8080	Enabled	Enabled	Unlc
8	Generic - Default Disk	/hpss_disk1	Enabled	Enabled	Unlc

At the bottom of the window, there are buttons for "Freeze", "Refresh", and "Dismiss". On the right side, there is a sidebar with several sections of buttons: "Device Administration" (Lock, Unlock, Mark Repaired), "Drive Administration" (Lock, Unlock, Mark Repaired, Dismount Drive), "Information" (Device Info, Drive Info), "Configuration" (Create Disk, Create Tape, Create Object Store, Configure, Delete), and "Preferences" (Edit, Default dropdown).

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.

### 14.1.2. 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.

### 14.1.3. 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 (see [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 (see [Mover Device Information window](#)) instead of the configuration window. These changes stay in effect until the Mover is recycled.

#### 14.1.4. 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 [Devices and Drives window](#) 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 [Devices and Drives window](#)).

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.

## 14.2. 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 [Devices and Drives window](#)). 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.

### 14.2.1. Mover Device Information window

**Mover Device Information**

File Edit Help

Basic Information

Device Name: /dev/tape/by-id/scsi-3500507630f316809

Device ID: 23

Device Type: IBM - 3592 E07/EH7 (TS1140) Tape

Mover: Mover (camille)

Administrative State: Unlocked

Operational State: ● Suspect

Usage State: ● Idle

Device Flags

Read Enabled     NO-DELAY Support     Removable Media Support  
 Write Enabled     TAOS Support     Locate Support  
 Fast Positioning     Disable RAO

State: [ ]

Set State: [ ]

Volume Flags: [ 0 ]

Number Of Errors: [ 1 ] [Reset](#)

Block Size: [ 0 ]

TaosFD: [ 0 ]

Volume ID: [ ]

LBPFlags: [ 0 ]

Bytes Read: [ 2,148,532,224 ] [Reset](#)

Bytes Written: [ 0 ] [Reset](#)

Freeze    [Update](#)    [Refresh](#)    [Dismiss](#)

Retrieving data - Succeeded

**Mover Device Information (on nudge.hpsslab.ibm.com)** x

**File Edit** **Help**

---

**Basic Information**

Device Name:

Device ID:

Device Type: **Generic - Default Disk**

Media Block Size:

Capacity:

Starting Offset:

Mover:

Administrative State:

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:

Number Of Errors:  [Reset](#)

Block Size:

TaosFD:

Volume ID:

LBPFlags:

Bytes Read:  [Reset](#)

Bytes Written:  [Reset](#)

---

Freeze  
 [Update](#)  
 [Refresh](#)  
 [Dismiss](#)

---

Retrieving data - Succeeded

**Mover Device Information (on linuxd2)**

File Edit Help

---

**Basic Information**

Device Name

Device ID

Device Type **AWS - S3 Object Store**

Mover

Administrative State

Operational State  Enabled

Usage State  Idle

---

**Device Flags**

Read Enabled  Write Enabled  SAN3P-transfer Enabled  Multiple Mover Tasks

---

State

Set State

Volume Flags

Number Of Errors

Block Size

TaosFD

Volume ID

LBPFlags

Bytes Read

Bytes Written

---

Freeze

---

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*

#### [What is TAOS?](#)



*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` 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.

### 14.2.2. PVL Drive Information window

The screenshot shows a window titled "PVL Drive Information" with a menu bar containing "File", "Edit", and "Help". The main area contains the following fields and controls:

Drive ID	8
Drive Type	Generic - Default Disk
Mover	Mover (malibu)
Administrative State	Unlocked
Operational State	Enabled
Usage State	Busy
Drive State	In Use
Drive Flag	Clear
Controller ID	8
Mounted Volume	MA000100
Maintenance Date	(ex. Jan 31, 2009 10:59:30 AM)
Drive Error Count	0 <input type="button" value="Reset"/>
Mounts Since Last Maintenance	26 <input type="button" value="Reset"/>
Comment	

At the bottom of the window, there is a "Freeze" checkbox and buttons for "Update", "Refresh", and "Dismiss". A status bar at the very bottom displays the message "Retrieving data - Succeeded".



**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 <a href="#">Reset</a>
Mounts Since Last Maintenance	0 <a href="#">Reset</a>
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 <a href="#">Reset</a>
Mounts Since Last Maintenance	495 <a href="#">Reset</a>
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).

## 14.3. 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.

### 14.3.1. 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 (see [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**.

### 14.3.2. Locking a drive

To lock a drive, go to the *Devices and Drives* window (see [Devices and Drives window](#)) 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.

### 14.3.3. 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.

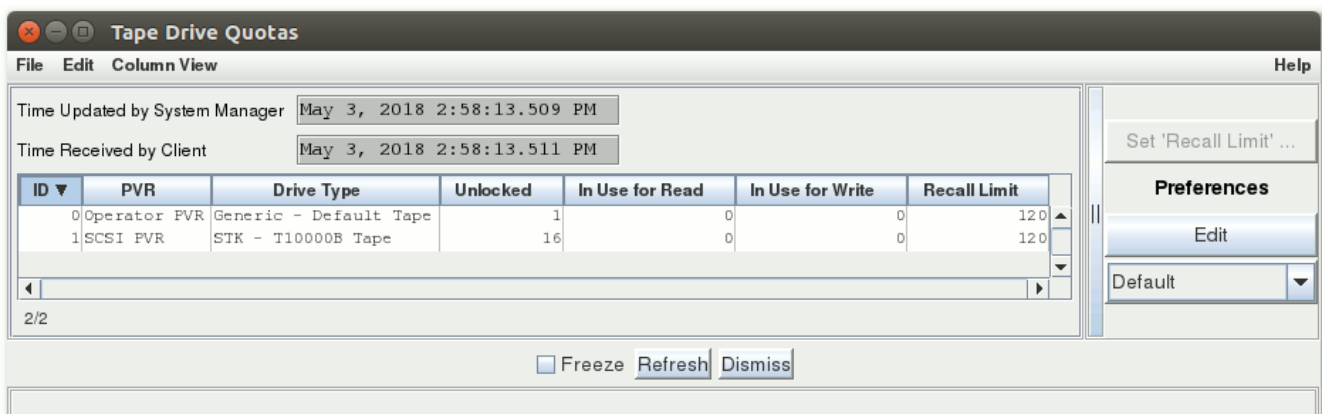
From the *Devices and Drives* window (see [Devices and Drives window](#)) select the desired device/drive entries and then click on the **Mark Repaired** button appropriate for that device/drive type. Another way to change a device/drive state to "repaired" is to bring up the *PVL Drive Information* 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.

### 14.3.4. 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.

### 14.3.5. 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 15. 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 chapter [Storage configuration](#).

for the procedures for creating these structures. \* Device configurations. See the chapter [Device and drive management](#).

for the procedures for creating these structures.

## 15.1. 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.

### 15.1.1. 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 (ACSL) 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 ([Import Disk Volumes window](#)).

, the *Import Object Store Volumes Window*, or the *Import Tape Volumes window* in the following section). Once the volumes are imported, the HPSS storage space can be created using the *Create Disk Resources* or *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.

#### **15.1.1.1. Import Tape Volumes window**

Import Tape Volumes

File Edit Help

PVR

Max Number of Drives

Manufacturer

Lot Number

Defer Labeling

Media Type

Import Type

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

Fill Increment

Volume Label

Volume List

Maximum Volumes Allowed

Total Count

Volume

Clear All Clear Selected Import Dismiss

Setting field: Media Type

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** 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.

#### 15.1.1.2. 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

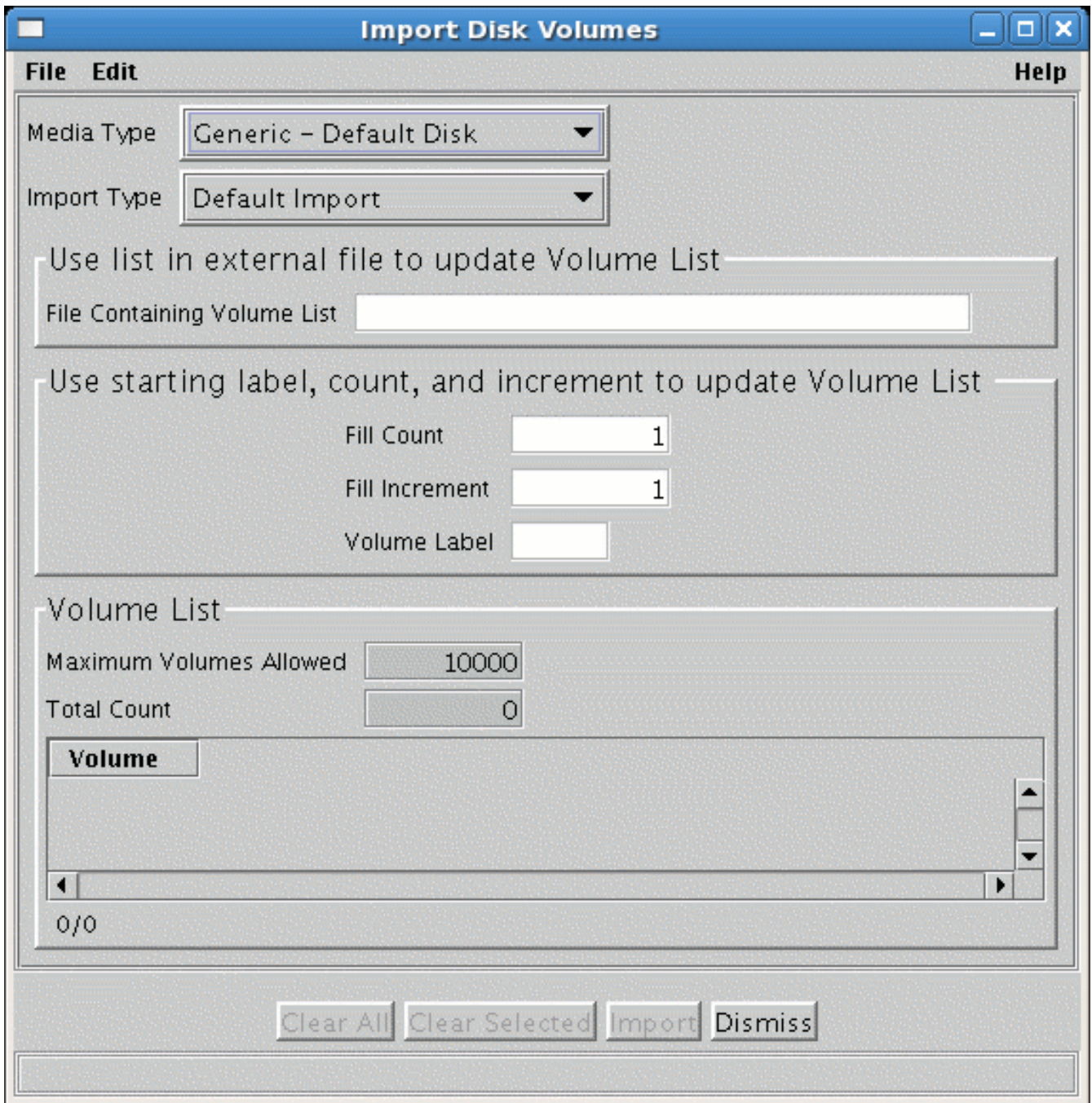
<b>Current tape label</b>	<b>Default import</b>	<b>Overwrite import</b>	<b>Scratch import</b>
An ANSI or HPSS label with an incorrect Volume ID (the Volume ID on the label is different from the Volume ID expected by HPSS)	Tape not imported	Tape not imported	Tape not imported
Random data (for example, 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.

### 15.1.1.3. 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** 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.

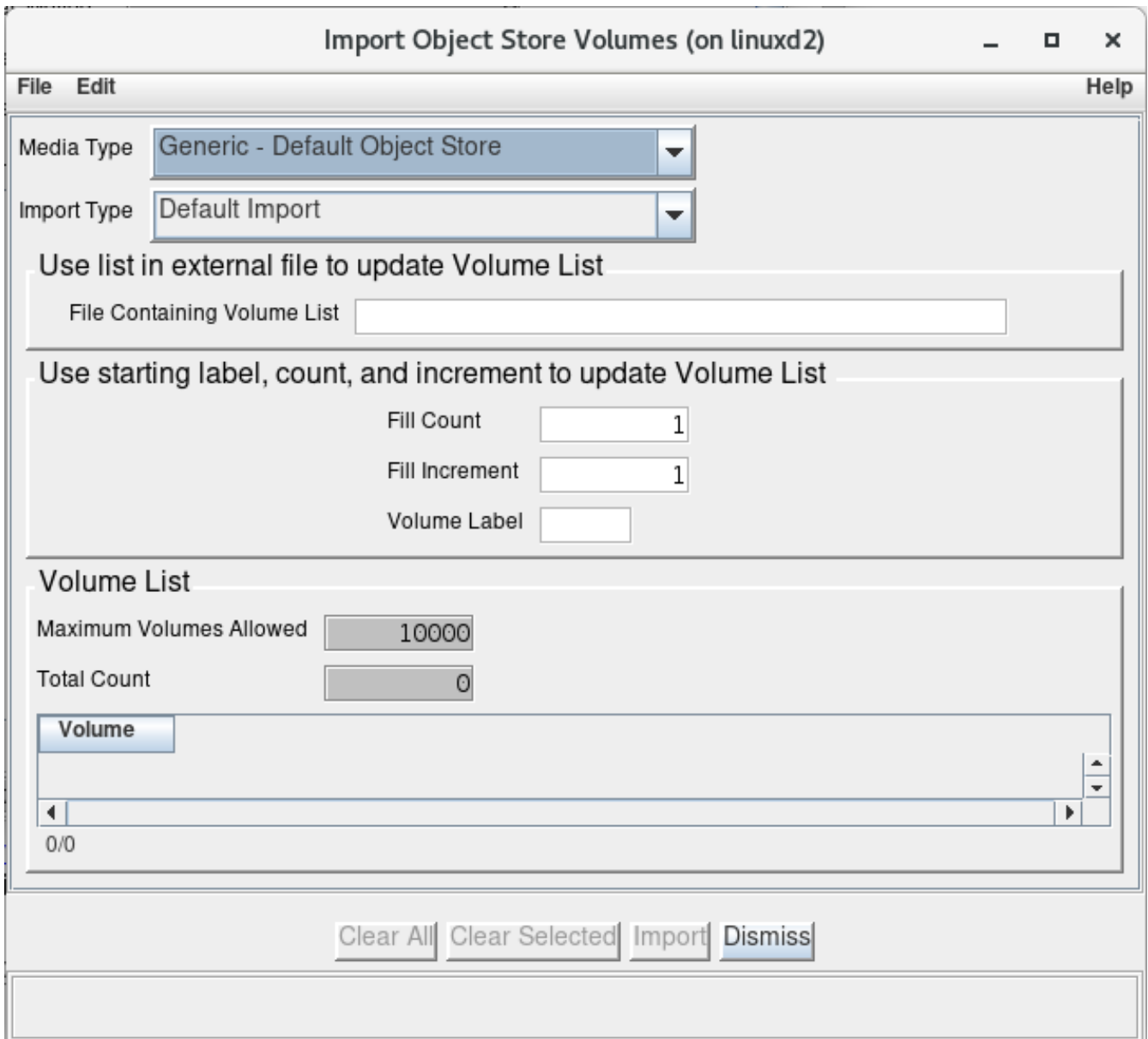
#### 15.1.1.4. Selecting import type for disk volumes

The following table lists information for selecting disk import types.

Table 29. Disk import types

<b>Current disk label</b>	<b>Default import</b>	<b>Overwrite import</b>	<b>Scratch import</b>
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

#### 15.1.1.5. 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** 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**. NOTE .Advice

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.

## 15.1.2. 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.

#### **15.1.2.1. 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 of 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** 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.

### 15.1.2.2. 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` tool

with the `-P` option, or by using the `hpss_vv_select_priority` 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 admin 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` 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.

See the man pages for `hpss_vv_select_priority`, `hpss_create_default_vv_select_weights`, `hpss_update_priority_by_pvr`, and `hpss_update_priority_by_pvr_drive` for more information.

### 15.1.2.3. 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` 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.

- 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.

#### 15.1.2.4. Create Disk Resources window

**Initialization Fields - Fill These In First**

Storage Class: 1000 | Storage Class 1000

Core Server: Core Server

**Optional or Informational Fields**

PVs in Each VV: 1

Fill Direction: Horizontal

PV Size: 42GB

Create DOWN

**Use list in external file to update Volume List**

File Containing Volume List: [Empty]

**Use starting label, count, and increment to update Volume List**

Fill Count: 1

Fill Increment: 1

Volume Label: DISK42

**Volume List**

Total Count: 1

VVs To Create: 1

Volume
DISK4200

1/1

Clear All Create Resources Dismiss

Setting field: Volume Label

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** 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.

#### **15.1.2.5. Create Object Store Resources window**

**Create Object Store Resources (on linuxd2)**

File Edit Help

**Initialization Fields - Fill These In First**

Storage Class 2 | FreeAWS (SC2) ▼

Core Server Core Server ▼

**Optional or Informational Fields**

Object Store Size 100PB

Create DOWN

**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 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** 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.

## 15.2. 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.

### 15.2.1. Deleting storage resources

Storage resources can be removed from HPSS volumes by the use of the *Delete Resources* window or the **remove** 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.

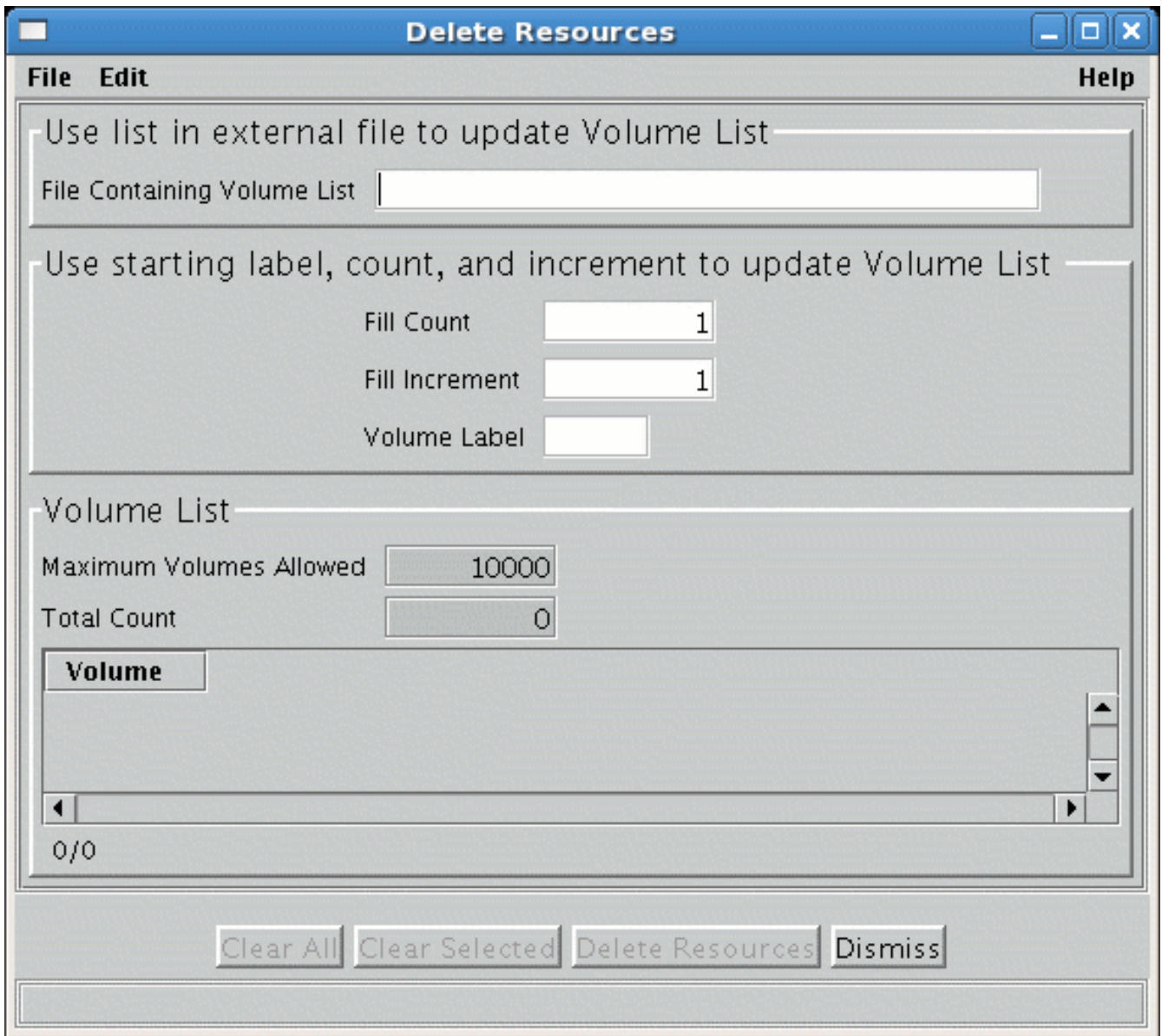
#### 15.2.1.1. Rules for deleting resources

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** 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.

#### 15.2.1.2. 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](#)

## 15.2.2. Exporting volumes from HPSS

Exporting volumes from HPSS is the process of removing tape cartridges, disk volumes, or object store volumes from HPSS control.

### 15.2.2.1. 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:

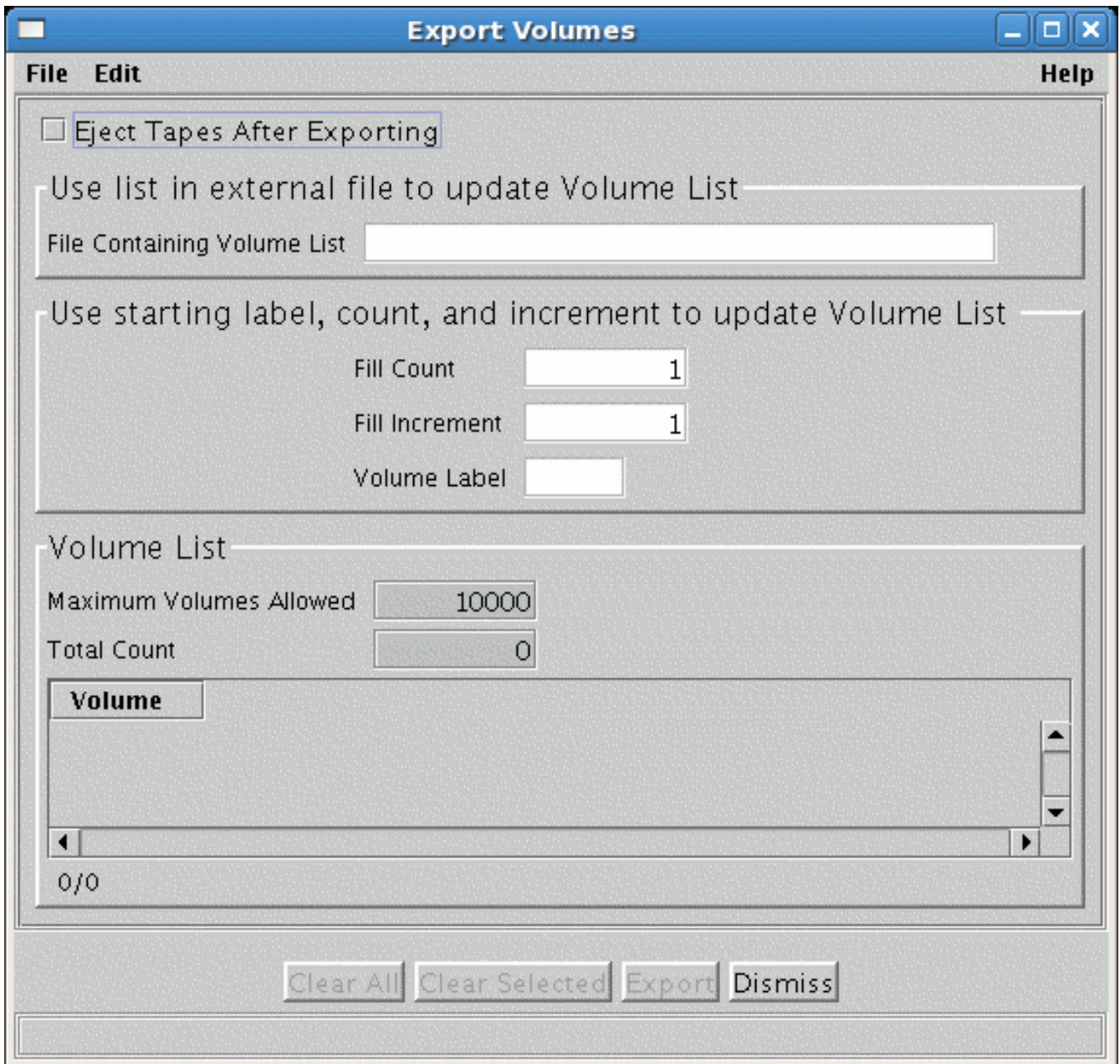
- **Operator** - No cartridge ejection step is necessary (or possible).
- **SCSI** - When an HPSS tape export command is issued, the cartridges will be placed in the input/output (I/O) slot and can then be manually removed.
- **STK** - The exported cartridges will be placed in the CAP and can then be 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.

### 15.2.2.2. 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** 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.

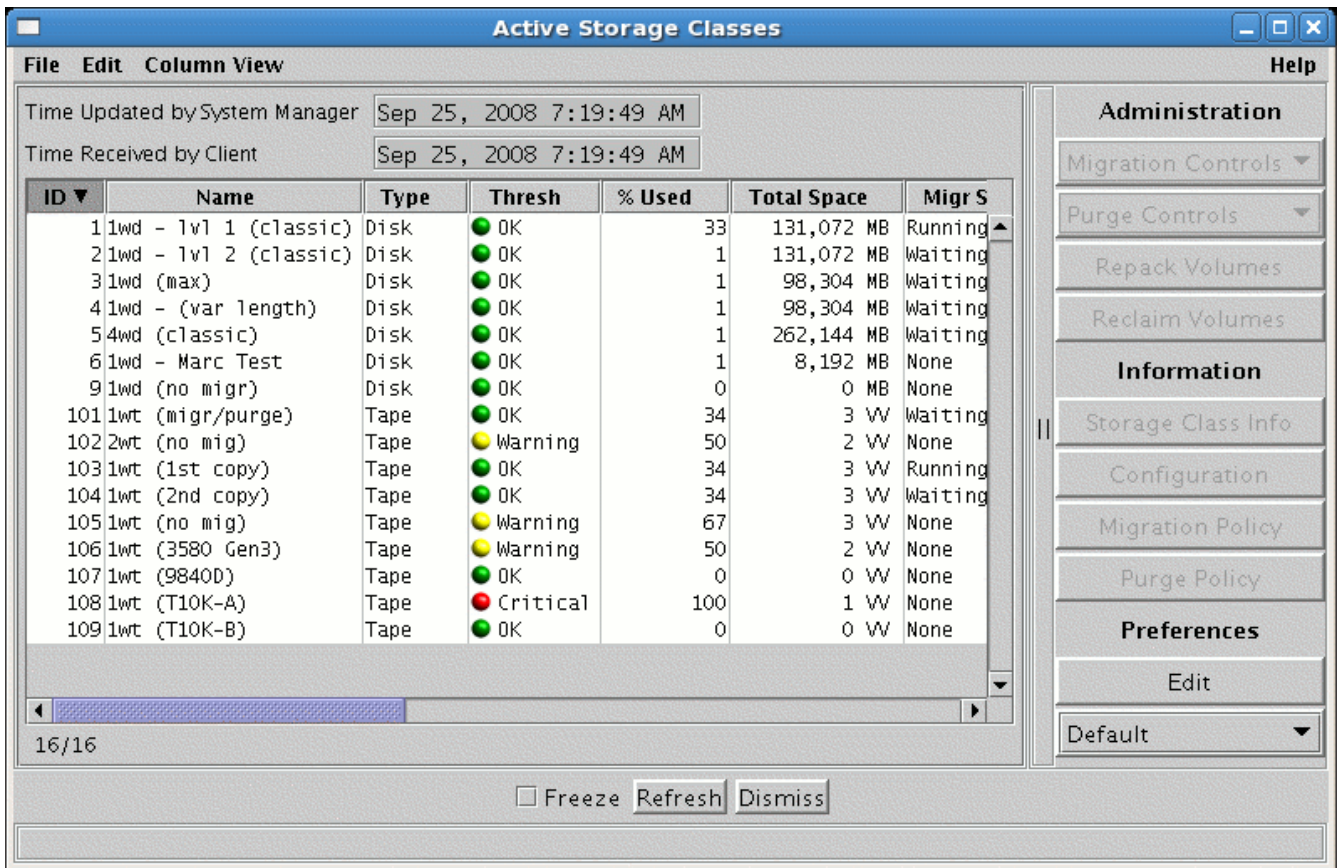
## **15.3. Monitoring storage space**

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.

### **15.3.1. 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** 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** 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.

## 15.3.2. MPS Disk Storage Class Information

**MPS Disk Storage Class Information** [ - ] [ □ ] [ X ]

**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        

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	<input type="text" value="Storage Class 1"/>	
Storage Class ID	<input type="text" value="1"/>	
Storage Class Type	Disk	
Subsystem Name	<input type="text" value="Subsystem #1"/>	
Total Space	<input type="text" value="8,589,926,400"/>	bytes
Free Space	<input type="text" value="8,588,865,536"/>	bytes
Space Used	<input type="text" value="1"/>	percent
Warning Threshold	<input type="text" value="80"/>	percent
Critical Threshold	<input type="text" value="90"/>	percent
Threshold Status	<span style="color: green;">●</span> OK	

**Migration Attributes**
**Purge Attributes**

Policy Name	<input type="text" value="Migration Policy 3"/>	
Policy ID	<input type="text" value="3"/>	
State	Waiting	
Start Time	<input type="text" value="Sep 24, 2008 2:34:34 PM"/>	
End Time	<input type="text"/>	
Total Units Processed	<input type="text" value="0"/>	bytes
Control	<input style="border: 1px solid gray;" type="text" value="None"/>	
Pending Operations	<input type="text"/>	

Freeze

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.

### 15.3.3. MPS Tape Storage Class Information

**MPS Tape Storage Class Information** [ - ] [ □ ] [ X ]

**File Edit Help**

Storage Class Name:

Storage Class ID:

Storage Class Type:

Subsystem Name:

Total Space:  VVs

Free Space:  VVs

Partially Filled VVs:

Space Used:  percent

Warning Threshold:  volumes

Critical Threshold:  volumes

Threshold Status: ● Critical

**Migration Attributes**

Policy Name:

Policy ID:

State:

Start Time:

End Time:

Total Units Processed:

Control:  ▼

Pending Operations:

Freeze

Retrieving data - Succeeded

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.

## 15.3.4. MPS Object Store Storage Class Information

**MPS Object Store Storage Class Information (on linuxd2)** ×

**File**   **Edit** **Help**

---

Storage Class Name

Storage Class ID

Storage Class Type **Object Store**

Subsystem Name

Total Bytes  bytes

Free Bytes  bytes

Space Used  percent

Warning Threshold  percent

Critical Threshold  percent

Threshold Status ● OK

---

Freeze        

---

Retrieving data - Succeeded



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**.

# 15.4. 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** program moves information from sparsely filled volumes to other volumes in the same storage class. When the **repack** 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** program, which refreshes the Core Server metadata that describes the volumes. See [Repacking and reclaiming volumes](#).

## 15.4.1. 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.

### 15.4.2. 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.

### 15.4.3. 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** 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** 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**.

**Repack** selects tape volumes in one of two ways. The administrator can provide a list of tape volumes to repack, or **repack** can select volumes based on a number of selection criteria. If **repack** 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** 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** 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** 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** utility program. This program can help to shed further light on whether an apparently degraded RAIT volume should be repacked.

Tape **repack** can segregate repack output tapes from other tapes. When this feature is enabled, **repack** 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** instance. This is controlled by **repack'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** man page.

When the **repack** 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** 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** 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** 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.

#### 15.4.4. 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.

##### 15.4.4.1. Repack Virtual Volumes window

Repack Virtual Volumes

File Edit Help

Storage Class Name

Storage Class ID

Subsystem ID

Core Server

File Family Criteria

Number of VVs to Repack

VV Space  percent

Max Bytes in Aggregate  (0 to disable file aggregation)

Estimated Size

**Repack Options**

Select Only Retired VVs

Include Shelved VVs

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** 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**.

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**. 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** 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** 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.

### 15.4.4.2. Reclaim Virtual Volumes window

The screenshot shows a window titled "Reclaim Virtual Volumes" with a standard Windows-style title bar (minimize, maximize, close buttons). Below the title bar is a menu bar with "File", "Edit", and "Help". The main content area contains five input fields:

- Storage Class Name: Tape T10KA
- Storage Class ID: 1000
- Subsystem ID: 1
- Core Server: Core Server
- Number of VVs to Reclaim: 1

At the bottom of the window, there are three buttons: "Reclaim", "Start Over", and "Dismiss".

This window is used to run the **reclaim** 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** 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.

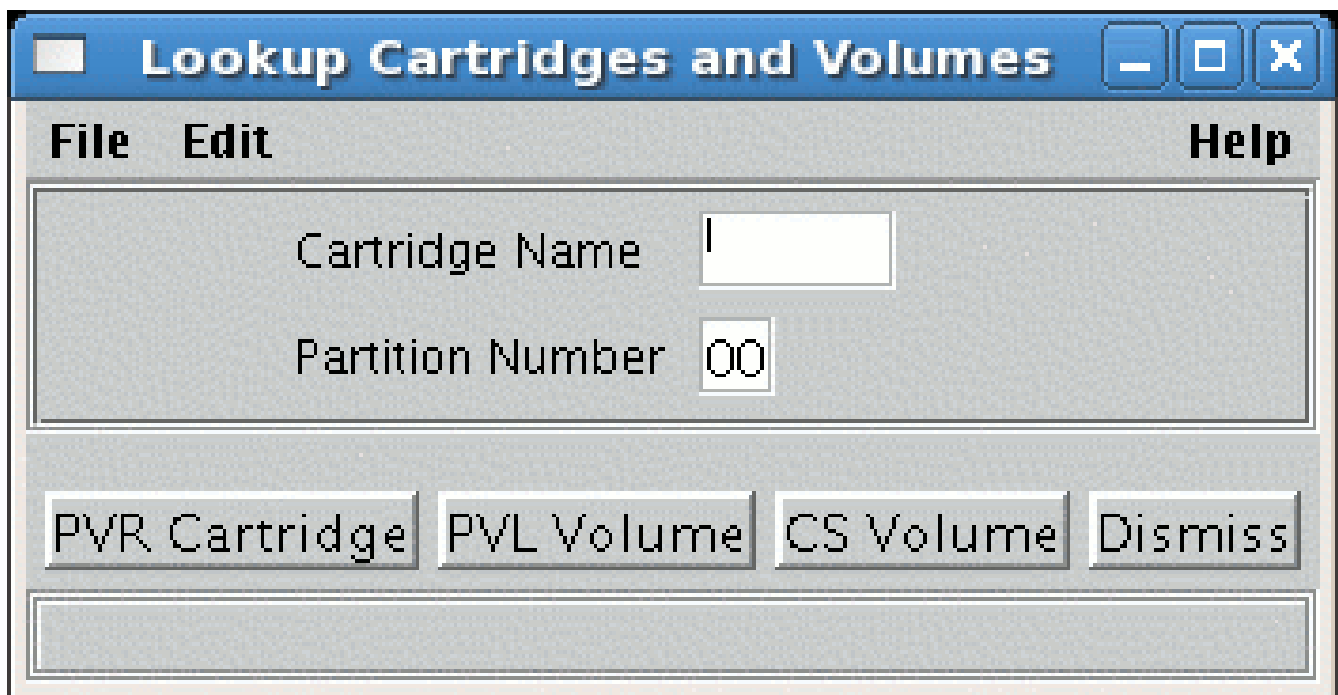
SSM invokes the **reclaim** utility program and passes it the storage class and number of tape volumes to be reclaimed, as entered on the *Reclaim Virtual Volumes* window. The utility program selects empty tape volumes from the storage class until it selects the required number or the supply of empty volumes is exhausted. The volumes are reclaimed and immediately become available for use.

The **reclaim** 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. Refer to the **reclaim** man page for more information on how to invoke the utility.

## 15.5. Volume management

This section describes how to monitor and manage PVL volumes, PVR cartridges, and Core Server disk and tape volumes.

### 15.5.1. 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.

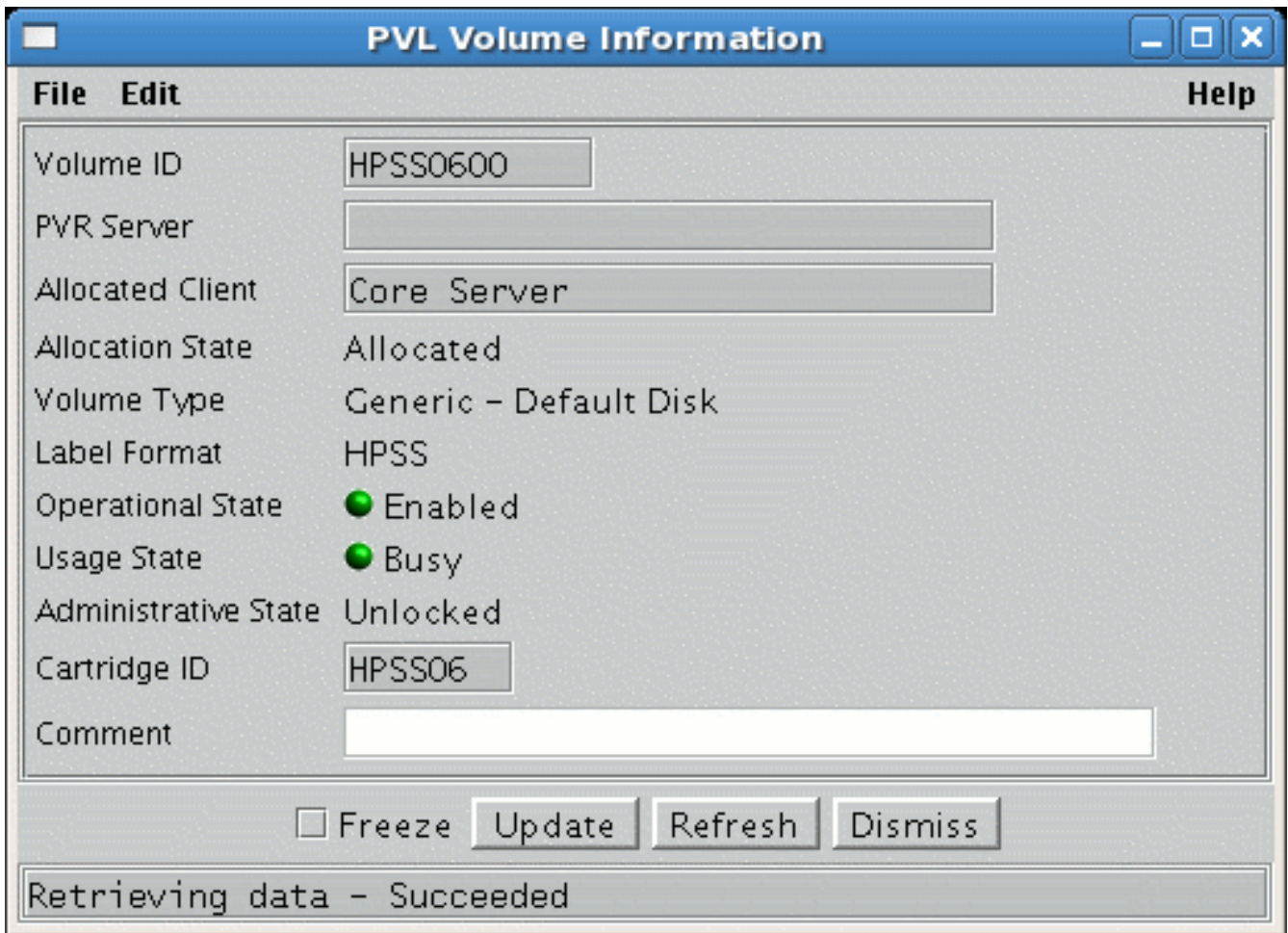
#### **CS Volume**

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.

## **15.5.2. 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 **dumpvv\_pvl** 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.

### 15.5.3. 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 **dumpvv\_pvr** utility can be used to list all the cartridges being managed by the PVR and help determine the label.

**PVR Cartridge Information**

File Edit Help

Cartridge ID: X02030

Sides: 1

Mounted Side: 0

PVR Server: Operator PVR

Cartridge Type: STK - T10000A Tape

Manufacturer:

Lot Number:

Service Start Date: May 15, 2008 10:10:56 AM

Maintenance Date: Oct 30, 2008 8:00:00 AM (ex. Jan 31, 2009 10:59:30 AM)

Last Mounted Date: May 15, 2008 11:25:23 AM

Mounts In Service: 1

Mounts Since Maintenance: 0

Operational State:  Enabled

Usage State:  Active

Administrative State: Unlocked

Mount Status:  Dismounted

Job Id: 0

Location

Type: None

Port: 0

Drive: 0

Slot

Unit: 0 Panel: 0 Row: 0 Column: 0

Freeze

Retrieving data - Succeeded

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** 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

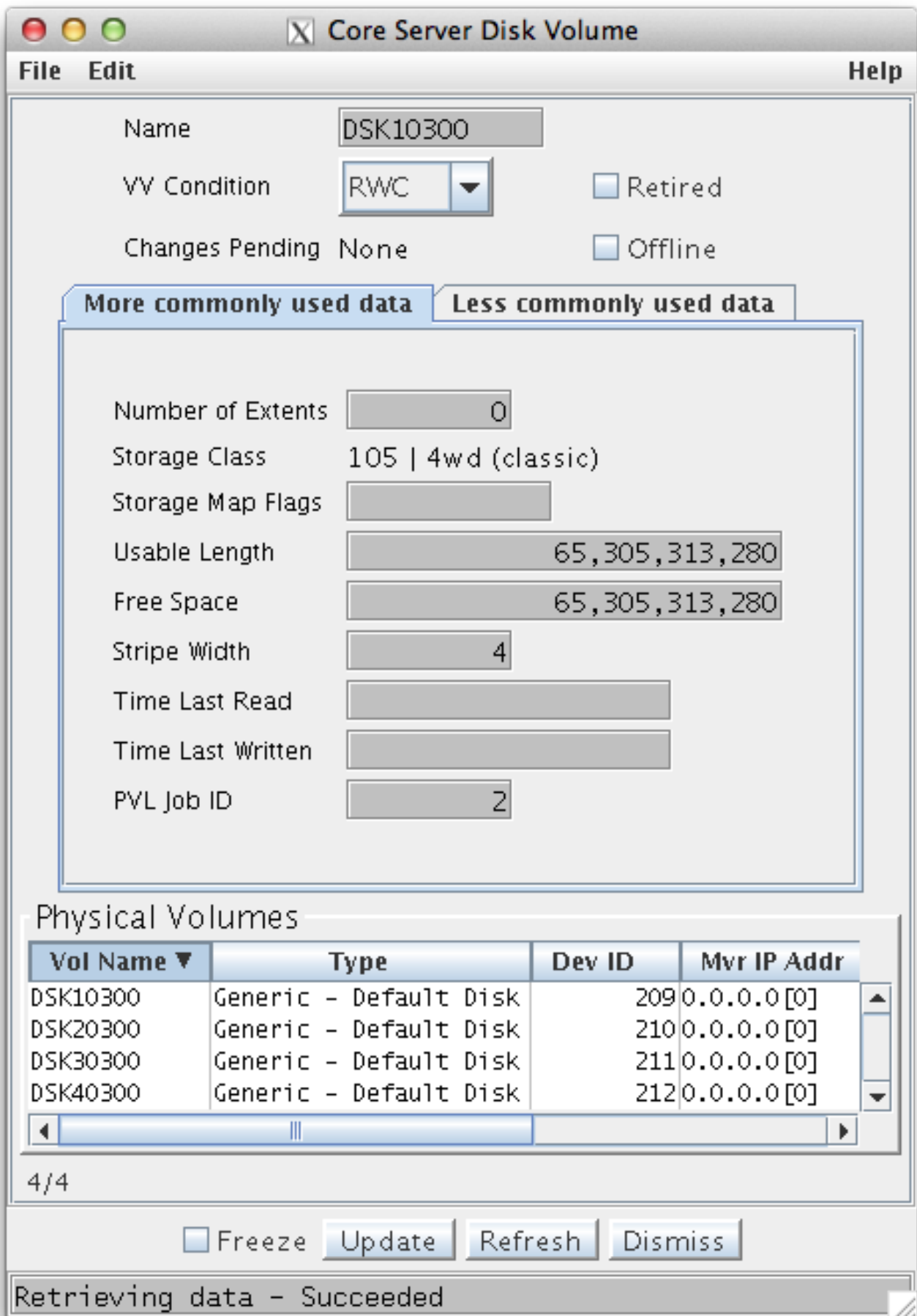
When **Mount Status** is displaying **Mount Pending**, the **Cancel Mount** button to the right of the field will be sensitized. If you click the button, a large warning will be displayed, and you will have to confirm your request by clicking **Cancel Mount** again. This will send a request to the PVR 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.*

## 15.5.4. Core Server Volume and Segment windows

### 15.5.4.1. 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 **hpssgui 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 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.

**Core Server Disk Volume**

File Edit Help

Name:

VV Condition:   Retired

Changes Pending:   Offline

**More commonly used data** | **Less commonly used data**

Actual Length:

PV Size:

Cluster Length:

VV BlockSize:

PV BlockSize:

Creation Time:

VV ID:

Object Id:

**Physical Volumes**

Vol Name ▼	Type	Dev ID	Mvr IP Addr	Mvr	Mvr
D0123300	Generic - Default Disk	2233	0.0.0.0 [0]	Mover (hpss-mvr20)	
D0123400	Generic - Default Disk	2234	0.0.0.0 [0]	Mover (hpss-mvr20)	
D0123500	Generic - Default Disk	2235	0.0.0.0 [0]	Mover (hpss-mvr20)	
D0123600	Generic - Default Disk	2236	0.0.0.0 [0]	Mover (hpss-mvr20)	

4/4

Freeze

Retrieving data - Succeeded

## 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.



### 15.5.4.2. Core Server Tape Volume information window

**Core Server Tape Volume**
\_ □ ×

File Edit
Help

Name

WV Condition   Retired

Changes Pending None  FIFO

More commonly used data
Less commonly used data
RAIT Options

Number of Active Segments

Map State Free

Storage Class 2 | 1wt (tape)

Map Flags

Max Written Length

Active Length

Estimated Size

Space Left

Number of Reads

Number of Writes

Time Last Read

Time Last Written

Time Last Verified

PVL Job ID

Current Position

Section

Offset

Next Write Address

Section

Offset

Physical Volumes

Vol Name ▼	Type	Dev ID	Condition	Errors	Format
37312000	IBM - 3592 E07 Standard Tape	0	FW		HI ▼

Freeze

Retrieving data - Succeeded

This window displays information about a tape volume as represented by the Core Server.

## 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** 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.

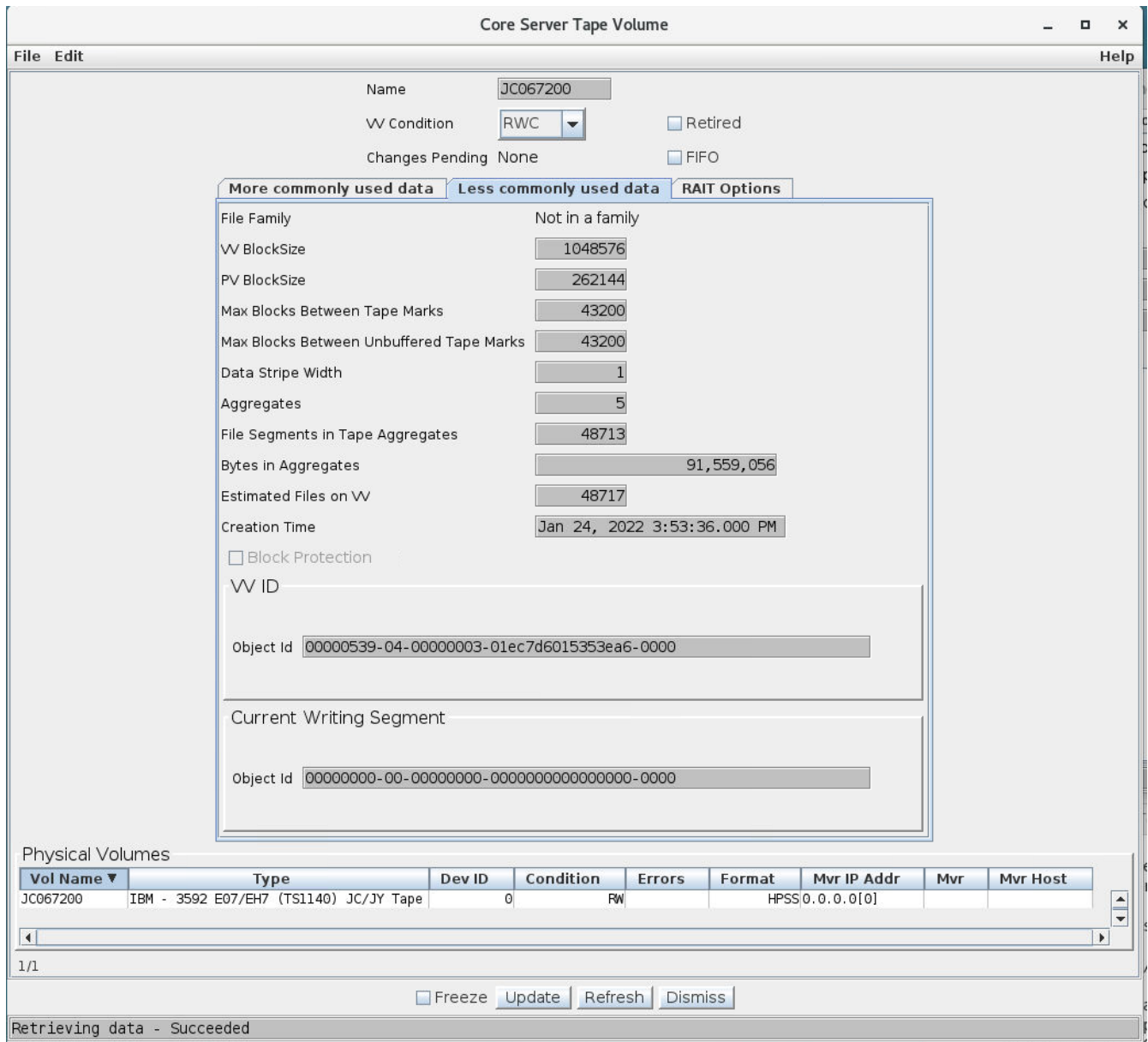
### **Current Position**

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.

#### WV 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 be 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**

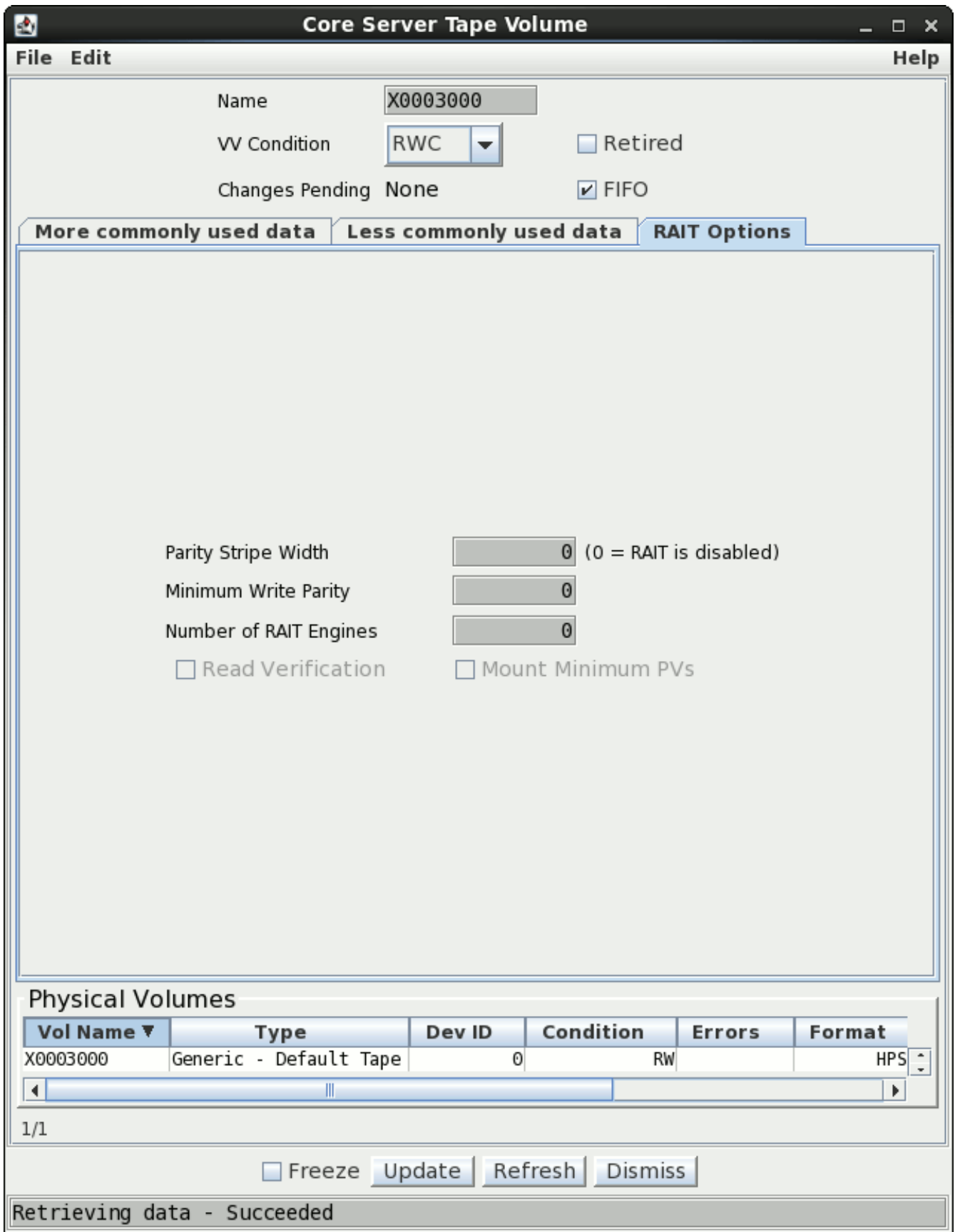
When this flag is set, Logical Block Protect (LBP) will be enabled for the physical tape volumes in this virtual tape volume. When LBP is enabled, HPSS will generate and pass a CRC with each block written to the tape volume and validate the CRC accompanying each block read from the tape volume.

**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 of 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** 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.

### 15.5.4.3. Core Server Object Store Volume information window

Core Server Object Store Volume (on linuxd2)

File Edit Help

Name: 00000700

VV Condition: RWC  Retired

Changes Pending: None  Offline

More commonly used data | Less commonly used data | Object store specific data

Storage Class: 6 | HPSSAWS (SC6)

Map Flags:

Time Last Read: May 10, 2024 10:25:14.000 AM

Time Last Written: May 10, 2024 10:16:04.000 AM

Files: 1

Bytes Used: 2,507,776

Number of Active Segments: 373

PVL Job ID: 5

Physical Volumes

Vol Name	Type	Dev ID	Mvr IP Addr	Mvr	Mvr Host
00000700	AWS - AWS S3 Object Store		70.0.0.0[0]	Mover (linuxd2)	

Freeze Update Refresh Dismiss

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 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 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.

Core Server Object Store Volume (on linuxd2)

File Edit Help

Name: 00000700

VV Condition: RWC  Retired

Changes Pending: None  Offline

More commonly used data | **Less commonly used data** | Object store specific data

Creation Time: Oct 9, 2023 9:18:16.000 AM

VV ID: [Empty]

Object Id: 0000d903-08-00000004-01ee66aeb025b88a-0000

Physical Volumes

Vol Name	Type	Dev ID	Mvr IP Addr	Mvr	Mvr Host
00000700	AWS - AWS S3 Object Store	70.0.0.0[0]	Mover (linuxd2)		

Freeze Update Refresh Dismiss

## 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.

Vol Name	Type	Dev ID	Mvr IP Addr	Mvr	Mvr Host
00000700	AWS - AWS S3 Object Store	70.0.0.0[0]	Mvr (linuxd2)		

### 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 flag indicates that the restore requests will register for notification to be sent to HPSS at completion of a restore.

### 15.5.5. 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.

### 15.5.6. 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.

### 15.5.6.1. Move Cartridges window

The screenshot shows the 'Move Cartridges to New PVR' window. The title bar includes standard window controls. The menu bar contains 'File', 'Edit', and 'Help'. The main interface is divided into several sections:

- New PVR:** A dropdown menu currently set to '3494 PVR'.
- Use list in external file to update Volume List:** A section containing a text input field labeled 'File Containing Volume List'.
- Use starting label, count, and increment to update Volume List:** A section containing three text input fields: 'Fill Count' (value: 1), 'Fill Increment' (value: 1), and 'Volume Label' (empty).
- Volume List:** A section containing two text input fields: 'Maximum Volumes Allowed' (value: 10000) and 'Total Count' (value: 0). Below these is a list box labeled 'Volume' which is currently empty. The list box has a scrollbar and a status indicator '0/0' at the bottom left.
- Buttons:** At the bottom of the window, there are four buttons: 'Clear All', 'Clear Selected', 'Move Cartridges', and 'Dismiss'.

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.

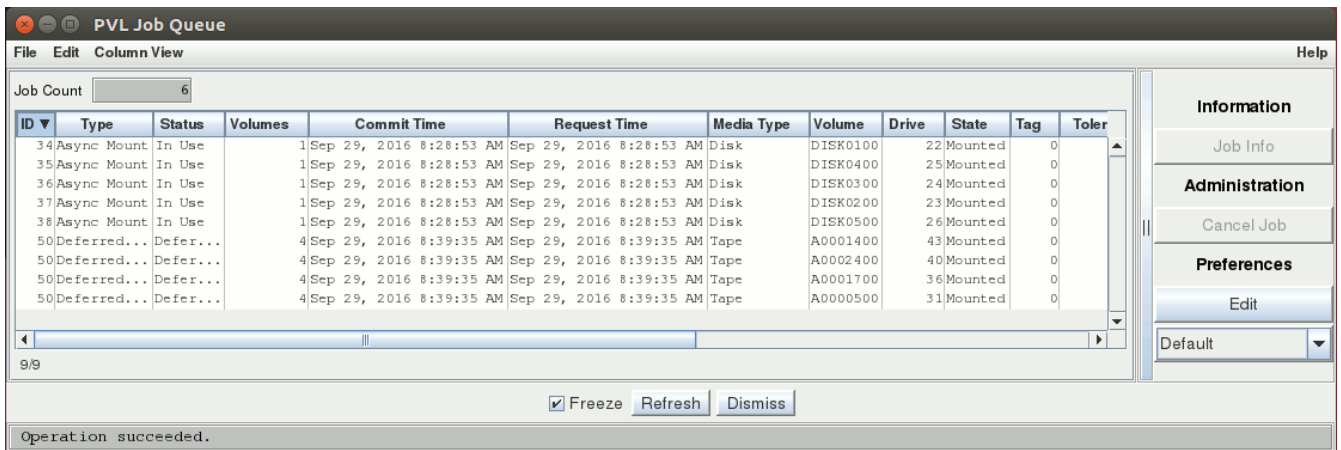
## **15.6. Monitoring and managing volume mounts**

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.

### **15.6.1. 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

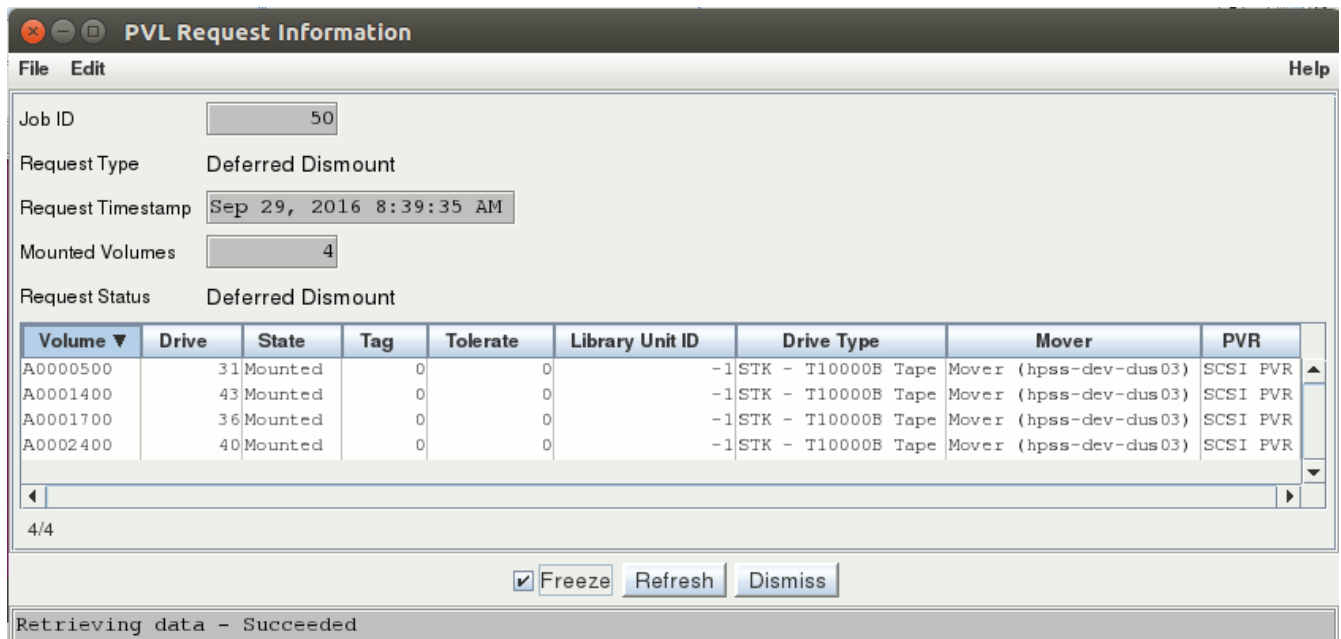
Click on this button to cancel the selected job. You are prompted to confirm your request before it is actually executed. This will generate a dismount request.

### Related Information

[Tape Mount Requests window](#)

[Canceling queued PVL requests](#)

## 15.6.2. 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.

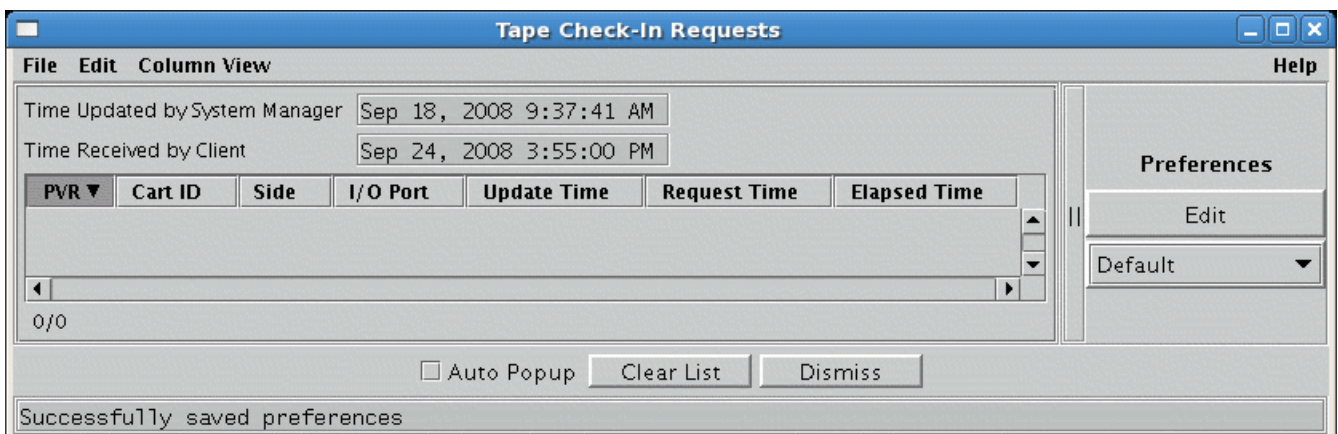
## 15.6.3. Canceling queued PVL requests

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. Cancellation 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.

## 15.6.4. 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** 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.

## 15.6.5. 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.

## **15.6.6. 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.

## 15.7. 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.

+ TIP: 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.

+ . Click **Apply Config** on the *HPSS Health and Status* screen for the modifications to the storage class to take effect. . 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.

+ TIP: 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.

+ . Use the **retire** utility program to retire the prior old volumes. \* **View the \*retire** man page for details on how to use the program.

+ TIP: **dump\_sspvs** can be used to create a list of physical volumes for **retire** to process. Alternatively, if the number of volumes to retire is small, the **hpssgui** 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**).

+ . Use the **repack** 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** to monitor the **repack** progress. **View the repack man page for details on how to use the program.** . Use the **remove** 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** man page for details on how to use the program.



# Chapter 16. Logging and status

---

## 16.1. 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`) and command line session (`hpssadm`) 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` and `hpssadm` 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.

## 16.2. 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.

### 16.2.1. Creating a log policy

A new log policy can be created and managed using the *Logging Policy* window (see [Logging Policy configuration window](#)).

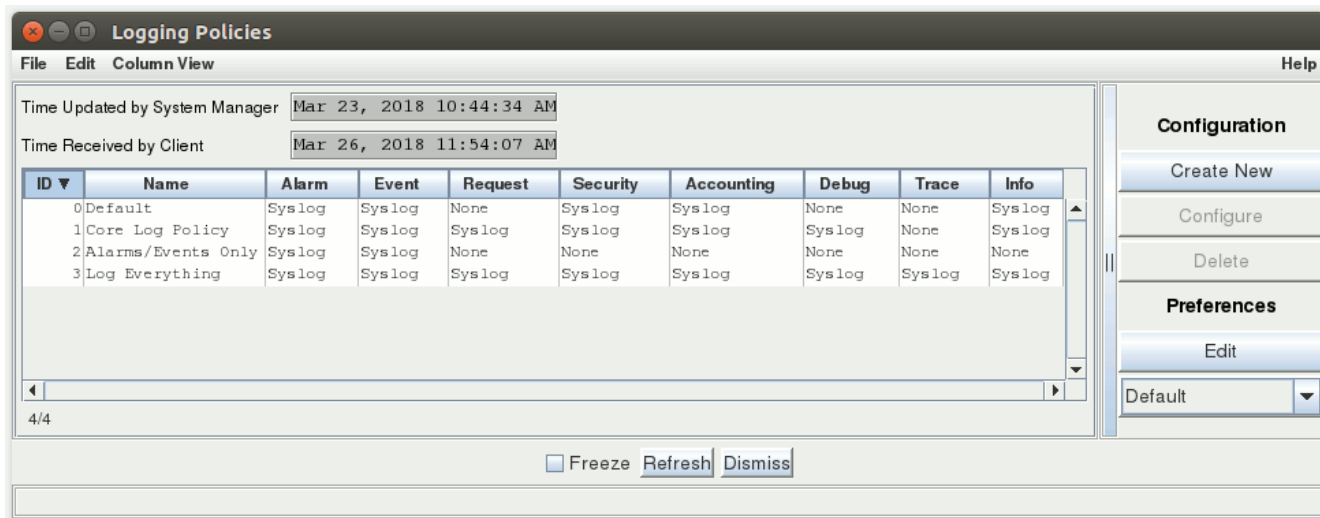
This *Logging Policy* window can be accessed from the *Logging Policies* window (see [Logging Policies window](#)).

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.

### 16.2.2. 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.

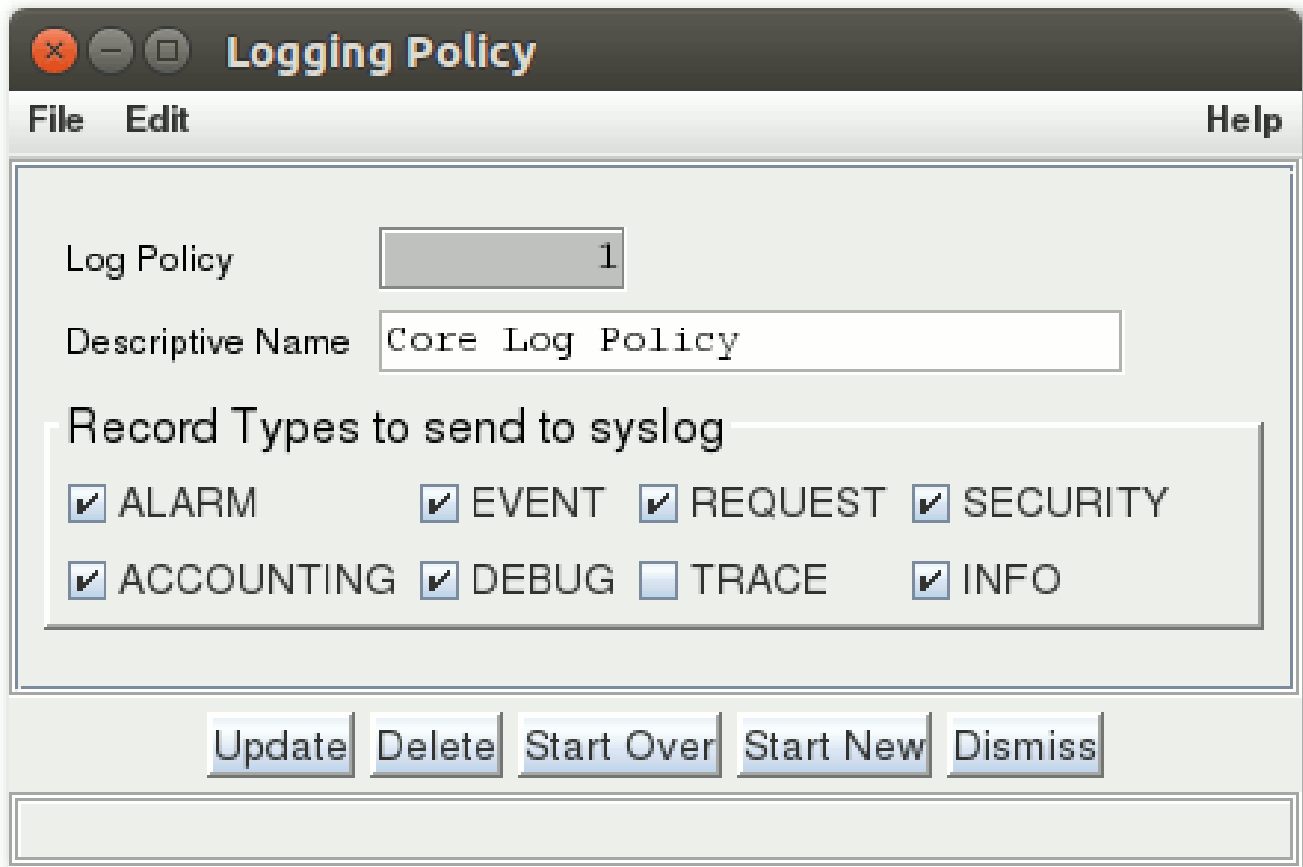
#### Configure

Opens the selected log policies for configuration editing.

#### Delete

Deletes the selected log policies.

#### 16.2.2.1. 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.*

### 16.2.3. 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.

### 16.2.4. 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.

## 16.3. 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.

### 16.3.1. 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`.

### 16.3.2. 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`. Mirroring HPSS alarms and events into `/var/log/messages` may be desirable.

Given the relatively verbose nature of the HPSS logs, it may be better for the majority of the HPSS logs to go elsewhere. 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`

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.*

### 16.3.3. 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.

### 16.3.4. 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);
...
};
```

### 16.3.5. 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` 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.

### 16.3.6. 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
```

### 16.3.7. 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 begin 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
```

### 16.3.8. 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
```

### 16.3.9. 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

File Name Format	Location	File Status
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 **man hpss\_log\_archive** 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** 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** 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** tool manually. See the [hpss\\_log\\_archive.7](#) man page for usage.*

### 16.3.10. 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** 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` 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** tool has additional options to control the way that files are written into HPSS. See the **hpss\_log\_archive** man page for details.

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** 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** - 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** tool currently does not work with the `dateext` logrotate option. The `nodateext` option must be used for the **hpss\_log\_archive** tool to process rotated log files.

### 16.3.11. 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.

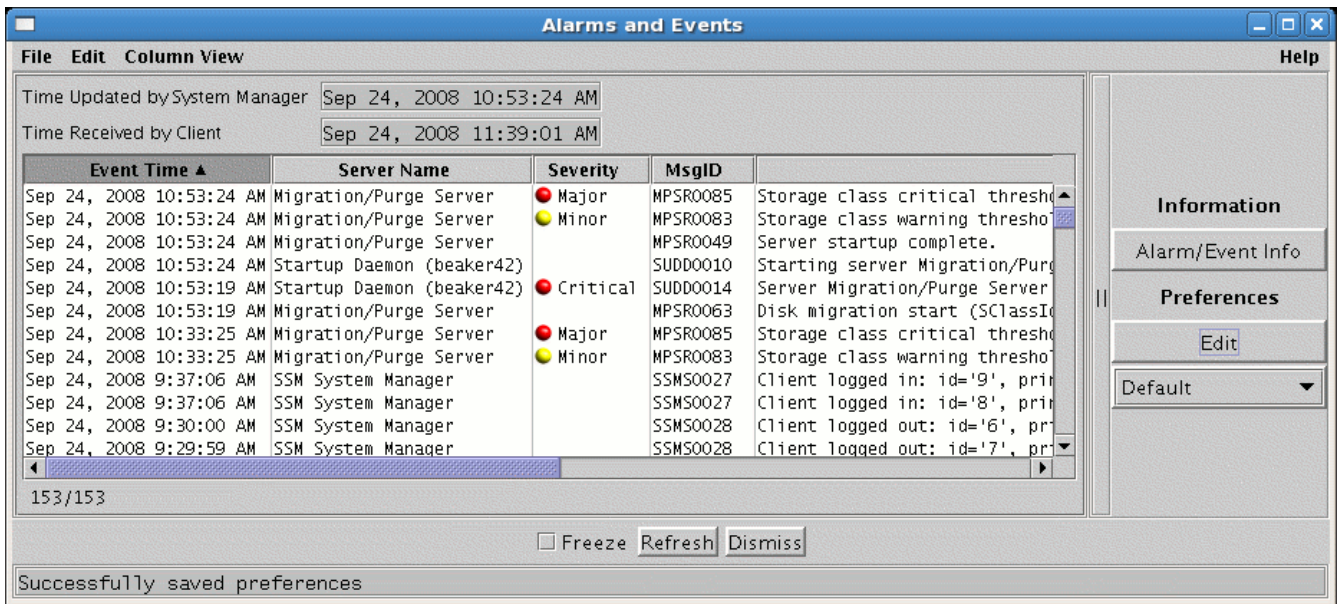
When the HPSS log file, referred to here as `logfile`, reaches the maximum size indicated by its logrotate policy, **logrotate** will rotate all already rotated logs (`logfile.1-logfile.N`) such that `logfile.1` becomes `logfile.2` and so on, and anything beyond the logrotate configured rotate count 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** with the file that was just rotated (`logfile.1`) and **hpss\_log\_archive** 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`.

## 16.4. Managing SSM alarms and events

This section describes how to manage SSM alarms and events.

## 16.4.1. Alarms and Events window

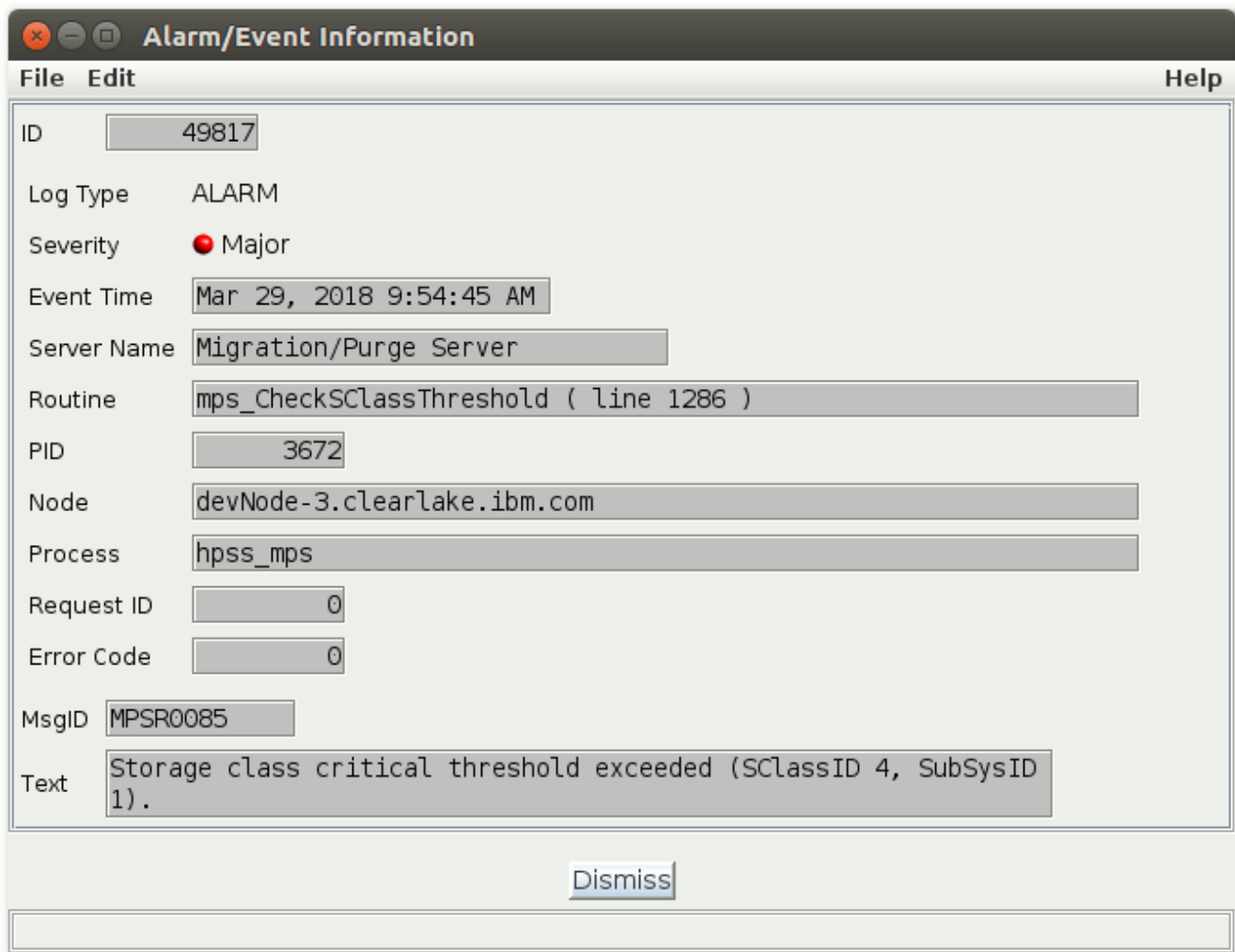


This window displays a number of the most recent alarm and event messages which have been received by SSM. It also allows you to view individual messages in greater detail by selecting the message and clicking the **Alarm/Event Info** button to bring up the *Alarm/Event Information* window.

### 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.

## 16.4.2. 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.

### 16.4.3. 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.

#### 16.4.4. 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** or **hpssadm** man page for a discussion of SSM session logging.

##### 16.4.4.1. 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.

#### 16.4.4.2. Controlling log messages displayed by **hpssgui** and **hpssadm**

The **hpssgui** and **hpssadm** 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** and **hpssadm** 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** or **hpssadm** 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** or **hpssadm** 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** or **hpssadm** polling interval and maximum number of messages values are not set appropriately, there can be gaps in the **hpssgui** or **hpssadm** 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** or **hpssadm** 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** 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** man page for details.

The internal cached alarm and event list is displayed by the **hpssgui** 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** 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** and the **hpssadm**, although the **hpssadm** 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** *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** user can filter and sort what is shown on his own *Alarms and Events* window independently of any other **hpssgui** 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** users.

### 16.4.5. 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.

## 16.5. 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/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_ftp.log:`**

PFTP uses `/var/hpss/ftp/adm/hpss_ftp.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/sqlldb/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`, 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 hexadecimal 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.

## 16.6. 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 `OperationsInProgress` for the mover(s) and VV(s) involved in the I/O. When the I/O is complete, the SS decrements the `OperationsInProgress` 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 `OperationsInProgress` 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 `OperationsInProgress` will be incremented.

The `TimeLastAllocated` 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 `OperationsInProgress` (4 reads; 4 writes) and **mover B** has 8 `OperationsInProgress` (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.

Metadata about the disk mover and VV I/O metrics — the most recent changes to the I/O metrics and the allocation decisions based upon them — can be viewed in the `/var/hpss/tmp/<PID>.diag`. And a point-in-time snapshot of the I/O metrics is available via the unsupported `showdiskmaps` command line tool.

The I/O metrics metadata is ephemeral. It ages out pretty quickly. The metadata is 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 metadata is 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 metadata and to control how long the server state dumps are, especially on busy production systems.

The 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. See [annotated example snippet below](#).
- A point-in-time snapshot of disk and mover I/O metrics. This is available in the `showdiskmaps` tool's output. 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

```

### 16.6.1. Sample Log of Disk I/O Operations

Here's a brief annotated snippet of the disk I/O operations log from a Core Server state dump taken soon after starting the Core Server. It shows some migrations (disk reads) and disk allocations.

```

===== Disk I/O Operations =====

# migration starts against mover 1041 & VV DLD21200

Start Disk I/O on Mover 1041

      08/16/2021 08:07:38 PM
      Request ID = e5eb0de439bd4043af33e5a790faac80
      VV = DLD21200
      Mover most recent allocation: 12/31/1969 04:00:00 PM
      VV most recent allocation: 12/31/1969 04:00:00 PM
      I/O Type = disk read
      I/O Operations in Progress:
                Prev Current
      Mover =      0      4
      VV =        0      4

# migration starts against mover 1029 & VV DLD20000

Start Disk I/O on Mover 1029

      08/16/2021 08:07:38 PM
      Request ID = 37175ef784bef741803e4738d56e3ac2

```

```

VV = DLD20000
Mover most recent allocation: 12/31/1969 04:00:00 PM
  VV most recent allocation: 12/31/1969 04:00:00 PM
I/O Type = disk read
I/O Operations in Progress:
          Prev Current
Mover =    0      4
  VV =    0      4

```

```

# disk allocation against mover 1033 & VV DLD20400
# - a few notes:
#   o mover 1029 is skipped: it's too busy due to the above migration
#   o mover 1041 is skipped: it's too busy due to the above migration

```

Allocate Space on Mover 1033

```

08/16/2021 08:08:29 PM
Request ID = 29f655dcfc72004ab6827da6c558449f
VV = DLD20400
Mover most recent allocation: 08/16/2021 08:08:29 PM
  VV most recent allocation: 08/16/2021 08:08:29 PM
I/O Type = disk write
I/O Operations in Progress:
          Prev Current
Mover =  -      0
  VV =  -      0

```

Movers/VVs considered for allocation:

```

Mover 1029:
| Ops=4 LastAlloc=12/31/1969 04:00:00 PM
└ too busy
Mover 1033:
| Ops=0 LastAlloc=12/31/1969 04:00:00 PM
└ selected for allocation
  VV DLD20400:
  | Ops=0 LastAlloc=12/31/1969 04:00:00 PM
  └ selected for allocation
Mover 1037:
| Ops=0 LastAlloc=12/31/1969 04:00:00 PM
└ used too recently
Mover 1041:
| Ops=4 LastAlloc=12/31/1969 04:00:00 PM
└ too busy

```

```

# disk allocation against mover 1037 & VV DLD20800
# - a few notes:
#   o the same busy movers above are still listed as busy (the migrations are
#     still running)
#   o now mover 1033 is also too busy because of the previous allocation
#   o so, the remaining mover, 1037, is selected for the allocation

```

## Allocate Space on Mover 1037

08/16/2021 08:08:29 PM

Request ID = 66fb03e3eefcab4c926ff025170445a0

VV = DLD20800

Mover most recent allocation: 08/16/2021 08:08:29 PM

VV most recent allocation: 08/16/2021 08:08:29 PM

I/O Type = disk write

I/O Operations in Progress:

	Prev	Current
Mover =	-	0
VV =	-	0

Movers/VVs considered for allocation:

Mover 1029:

| Ops=4 LastAlloc=12/31/1969 04:00:00 PM  
└ too busy

Mover 1033:

| Ops=0 LastAlloc=08/16/2021 08:08:29 PM  
└ used too recently

Mover 1037:

| Ops=0 LastAlloc=12/31/1969 04:00:00 PM  
└ selected for allocation

VV DLD20800:

| Ops=0 LastAlloc=12/31/1969 04:00:00 PM  
└ selected for allocation

Mover 1041:

| Ops=4 LastAlloc=12/31/1969 04:00:00 PM  
└ too busy

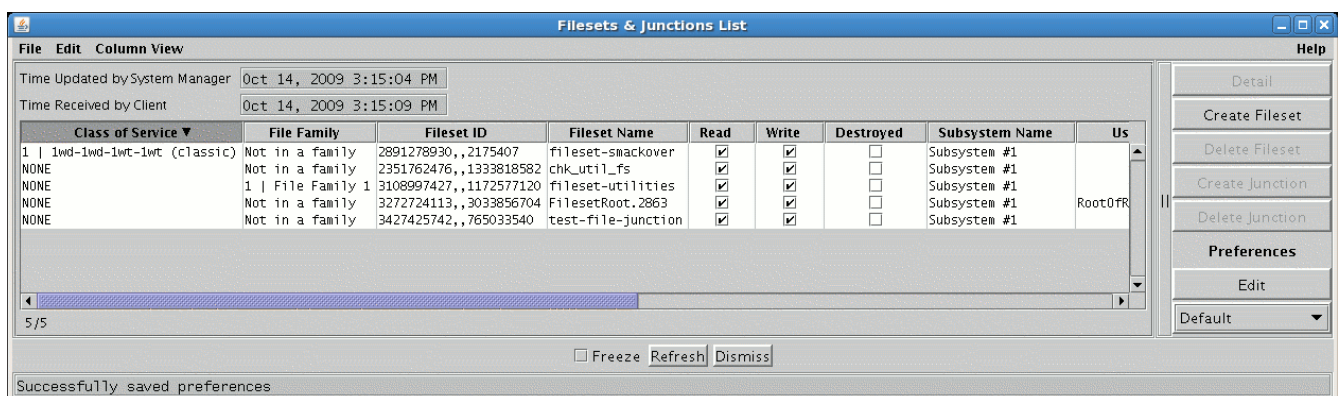
# Chapter 17. Filesets and junctions

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.

## 17.1. 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**, **lsjunctions** and **lshpss** 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.

## 17.2. 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** utility. Refer to the **crtfileset** man page for more information.*

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.



## 17.2.1. Create Fileset window

The screenshot shows the 'Create Fileset' dialog box. It includes a title bar with 'Create Fileset' and standard window controls. The main area is divided into several sections: 'Fileset Name' (text input), 'File Family' (dropdown), 'Class of Service' (dropdown), 'User Data' (text input), 'Core Server' (dropdown with a folder icon), 'UID' and 'GID' (text inputs), 'Fileset State' (checkboxes for 'Read' and 'Write'), and 'Permissions' (checkboxes for 'r', 'w', 'x' for 'User', 'Group', and 'Other'). At the bottom are 'Create' and 'Dismiss' buttons. A status bar at the very bottom says 'Setting field: Core Server'.

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.

## 17.3. Managing existing filesets

This section describes how to look up information on, modify, or delete filesets.

### 17.3.1. Core Server Fileset Information window

Core Server Fileset Information

File Edit Help

Fileset ID: 3272683715,,2360846864

Fileset Name: FilesetRoot.3202

Subsystem Name: Subsystem #1

File Family: Not in a family

Class of Service: NONE

User Data: Root0fRoots Fileset

Files: 49

Directories: 118

Sym Links: 0

Hard Links: 0

Junctions: 3

Core Server: Core Server

UID: 0

GID: 0

Permissions:

	r	w	x
User	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Group	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Other	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Fileset State:

Read

Write

Destroyed

Delete  Freeze Update Refresh Dismiss

Retrieving data - Succeeded

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.

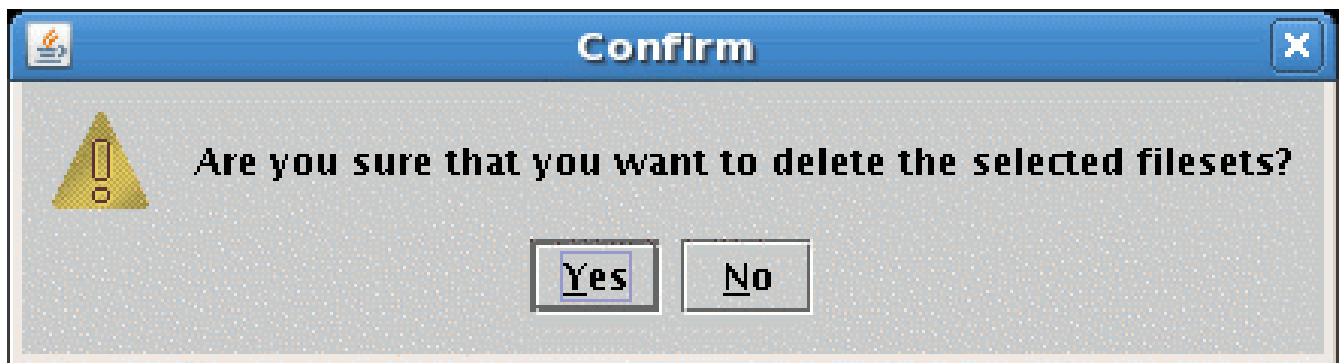
## Dismiss

Closes the window.

## 17.4. 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.

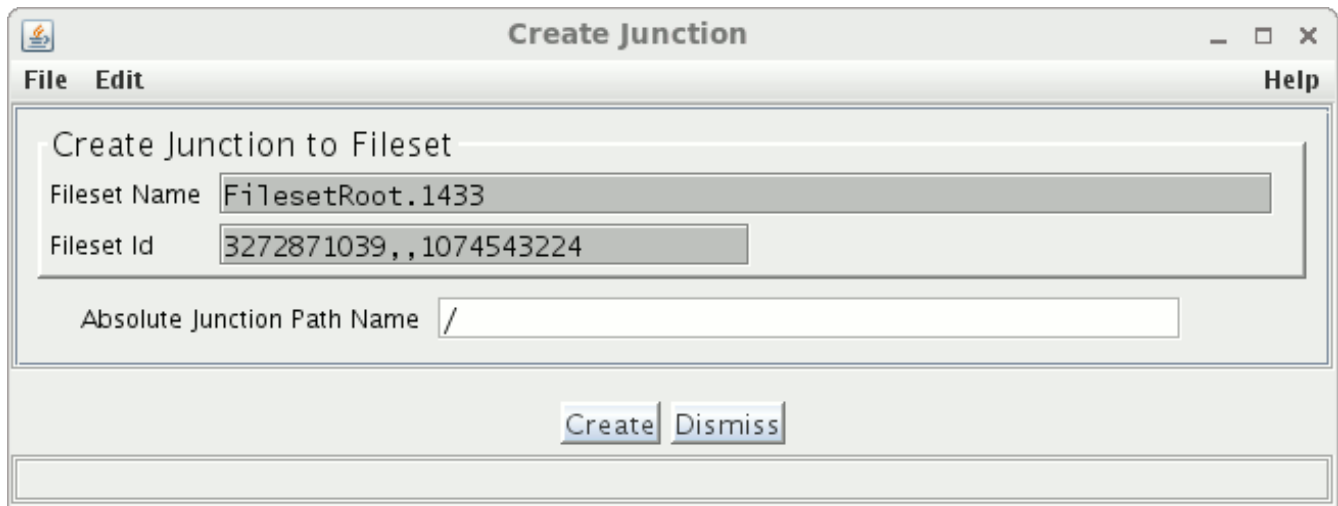
## 17.5. Creating a junction

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**. For more information, refer to the **crtjunction** man page.

### 17.5.1. Create Junction window



The screenshot shows a graphical user interface window titled "Create Junction". The window has a standard menu bar with "File", "Edit", and "Help" options. Below the menu bar, there is a title bar that reads "Create Junction to Fileset". The main content area contains three text input fields. The first field is labeled "Fileset Name" and contains the text "FilesetRoot.1433". The second field is labeled "Fileset Id" and contains the text "3272871039,,1074543224". The third field is labeled "Absolute Junction Path Name" and contains the text "/". At the bottom of the window, there are two buttons: "Create" and "Dismiss".

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.

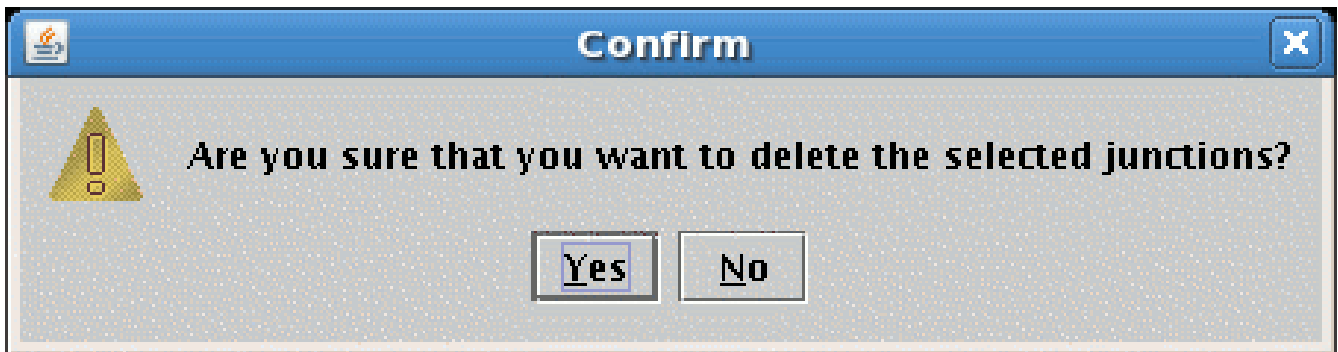
## Dismiss

Closes the window.

# 17.6. Deleting a junction

Junctions can be deleted using SSM or by using the utility routine **deljunction**. For more information, refer to the **deljunction** man page.

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**.

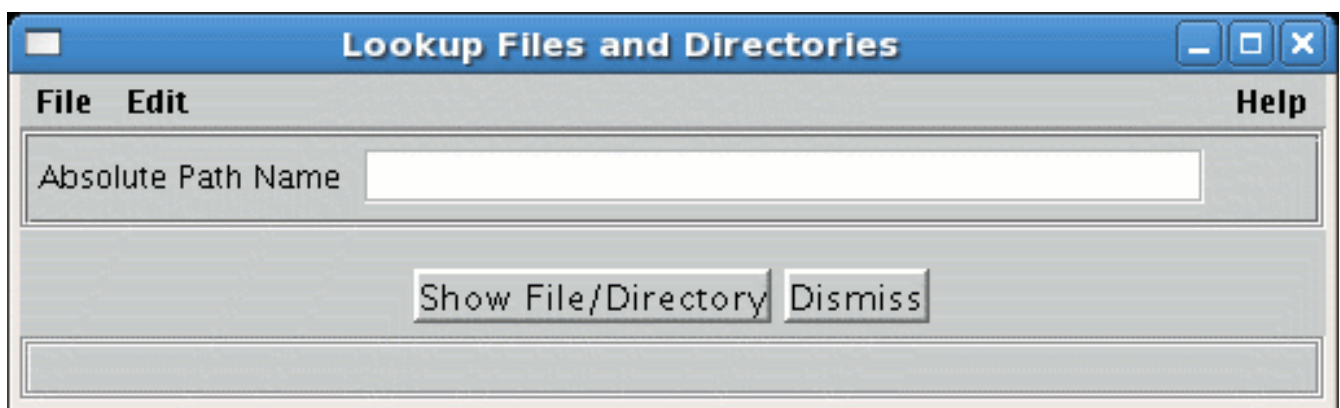


# Chapter 18. Files, directories and objects by SOID

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.

## 18.1. 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.

**18.1.1. File/Directory Information window**

File/Directory Information (on hpss-dev02.ccs.ornl.gov)

File Edit Help

Path Name

Object Type	File
Class of Service	1   Class of Service 1
File Family	Not in a family
Subsystem Name	<input type="text" value="Subsystem #1"/>
Realm Id	<input type="text" value="10000"/>
Account	<input type="text" value="0"/>
Read Count	<input type="text" value="0"/>
Write Count	<input type="text" value="0"/>
Link Count	<input type="text" value="1"/>
Creation Time	<input type="text" value="Nov 6, 2014 8:56:45 AM"/>
Modify Time	<input type="text" value="Nov 6, 2014 8:58:49 AM"/>
Last Written Time	<input type="text" value=""/>
Last Read Time	<input type="text" value="Nov 6, 2014 8:59:04 AM"/>
Data Length	<input type="text" value="0"/>
UID	<input type="text" value="0"/>
GID	<input type="text" value="0"/>
Permissions	<input type="text" value="rw-rw-r--"/>

Option Flags

Don't Purge

Extended ACL

Object ACL IC ACL IO ACL

Extended ACL

Fileset Information

Fileset Id

Fileset Root Object Id

Fileset State

Read  Write  Destroyed

Trashcan Information

Parent Id

User Id

Time Deleted

Time Created

Time Last Read

Time Modified

Original Pathname

Original Filename

Bitfile Id

Object Id

Hash Value   Valid

Handle to Object

Object Id	<input type="text" value="1,061,249"/>
Object Hash	<input type="text" value="9829"/>
File Id	<input type="text" value="1,061,249"/>
File Hash	<input type="text" value="9829"/>
Type	File
Flags	<input type="text" value="0x0"/>
Generation	<input type="text" value="8,215,264,866,564,785,059"/>
Core Server Id	<input type="text" value="13"/>

Freeze

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 is 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.

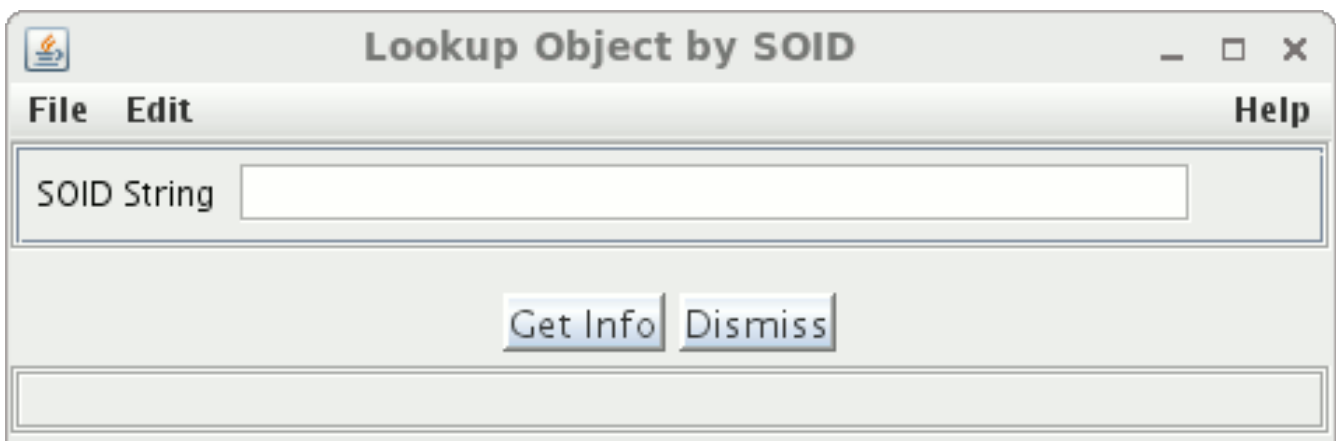
Note: The table below indicates which fields and attributes apply to both files and directories or to just one of the two. Refer to it to see which fields on the *File/Directory Information* window apply to files, directories, or both.

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	

## 18.2. 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 19. 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

## 19.1. 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.

## 19.2. 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.

## 19.3. 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.

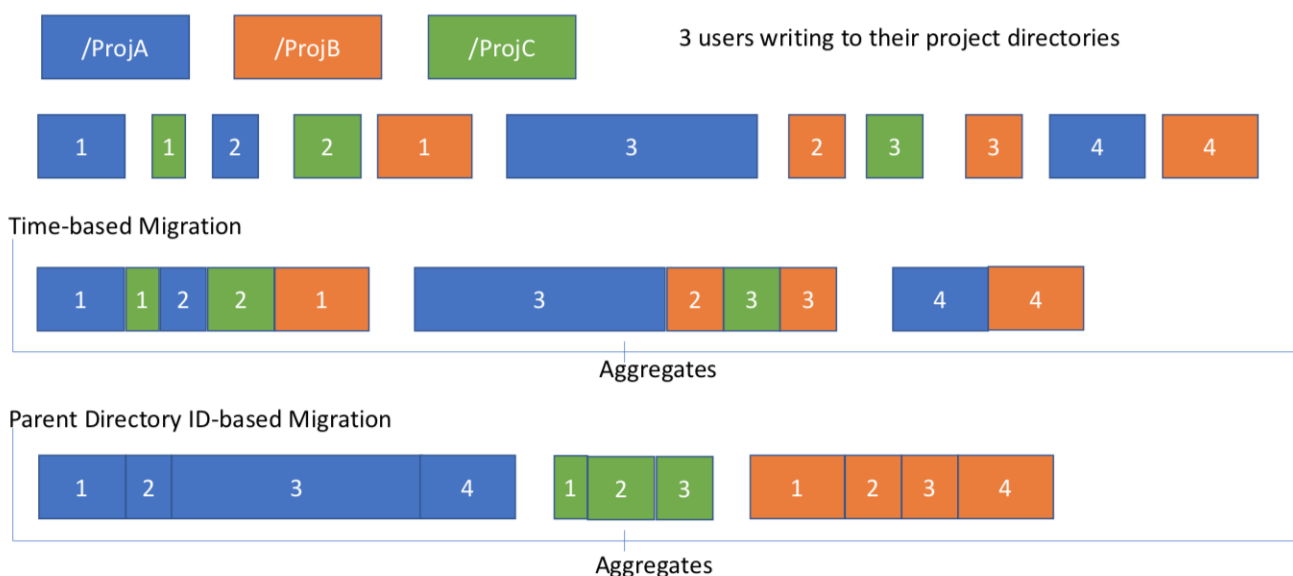
The objective of Ordered Migration is to co-locate data on tape, with either the directory of the files or the files' create times being the co-location criteria. A site may want to combine this feature with Full Aggregate Recall to potentially minimize the number of tape operations required to bring 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). This 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.

### 19.3.1. Ordered migration example - read-optimized aggregates



## 19.4. 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.

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 reinitialize 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.

## 19.5. 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 20. Tape Ordered Recall (TOR)

---

This chapter discusses the following operations:

- Overview of TOR
- TOR ordering behavior
- TOR performance considerations
- Configuring TOR
- Monitoring TOR
- Caveats

## 20.1. 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 enterprise tape 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 in a Virtual Volume Queue. 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** tool can take advantage of these APIs to facilitate mass recall from tape. See **man quaid** for details and usage.

Additionally, the **lori** tool makes use of ordered recall APIs similar to staging. The **lori** tool takes advantage of the read queueing APIs. See **man lori** for details and usage.

## 20.2. 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 certain enterprise drives, they will make use of a drive feature called Recommended Access Order (RAO). This 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. Ordering does not induce artificial wait times for queued items; if an item enters the queue as the sole item, it will immediately run. This means that, upon generating a set of items for recall, the first request to hit the queue will run without being ordered in with the other requests.

## **20.3. TOR performance considerations**

In order to get good performance with Tape Ordered Recall, there are several points to consider.

First, 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 requests queued up allows for better scheduling, which should result in less seek time and improved tape recall performance.

Second, HPSS has a number of configuration options which can impact the performance and behavior of TOR. The major configuration parameters which can affect TOR performance 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.

### **Thread Pool Size**

Same as above.

### Max Active I/O

The Core Server I/O scheduler has a limit to the number of I/O requests that are processed, subsystem-wide. 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".

### Max Active Copy

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 *Max Active I/O*, 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 10 pending stage requests, and the *Active Set Size* is set to "5", then the first five requests to enter the queue will be ordered and run. Then the next five 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.

### 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.

## 20.4. Configuring TOR

To configure Tape Ordered Recall, open the [Core Server-specific configuration](#) screen. Update the fields as necessary. The fields which directly affect TOR are *Tape Queue Method*, *Tape Queue Active Set Size*, and *Tape Max Reader Wait (seconds)*.

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 enterprise drive 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.

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`).

## 20.5. Caveats

1. Recommended Access Order currently has a hardware limit of 2730 requests that can be ordered. Sites using enterprise drives 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 an RAO-capable enterprise drive at 2730.

# Chapter 21. User accounts and accounting

## 21.1. 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**) 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** man page for more information on how to invoke the **hpssuser** utility.

### 21.1.1. Adding HPSS users

The **hpssuser** 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** utility can be invoked to simultaneously add multiple types of user IDs for a user, or to add an individual user type. Refer to the **hpssuser** man page for more information.



*Ensure that the Core Server is up and running before adding the user ID. The **hpssuser** utility will not be able to create the user's home directory if the Core Server is not available.*

#### 21.1.1.1. 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** 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** utility as follows to add the required user ID for an HPSS user:

```
hpssuser -add <user> -all
```

When invoked, the **hpssuser** 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 ]

```

### 21.1.1.2. Add a user ID

The **hpssuser** 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** utility will not be able to create the user's home directory if the Core Server is not available.

### 21.1.1.3. Add an SSM user ID

The **hpssuser** 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** 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 ]
```

### 21.1.2. Deleting HPSS users

The **hpssuser** 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 ]
```

### 21.1.3. Listing HPSS users

The **hpssuser** 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.*

#### 21.1.4. Create an SSM client package

The **hpssuser** 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 ]
```

## 21.2. 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.

### 21.2.1. Accounting Policy window

The screenshot shows the 'Accounting Policy' window with a menu bar containing 'File', 'Edit', and 'Help'. The main area contains the following fields:

- Accounting Style:** A dropdown menu with 'SITE' selected.
- Storage Unit Size:** A dropdown menu with 'Bytes' selected.
- Pathname of Executable (UNIX):** A text input field containing '/opt/hpss/bin/hpss\_acct'.
- Report File (UNIX):** A text input field containing '/var/hpss/acct/acct\_report'.
- Ordering:** A dropdown menu with 'Account, User, Group' selected.
- Options:** A group box containing three checkboxes: 'Account Validation', 'Require Default Account', and 'Account Inheritance', all of which are currently unchecked.

At the bottom of the window, there are four buttons: 'Update', 'Delete', 'Start Over', and 'Dismiss'.

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.

## **21.2.2. Accounting reports and status**

### **21.2.2.1. 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.

### 21.2.2.2. Accounting Status window

The screenshot shows a window titled "Accounting Status" with a menu bar containing "File", "Edit", and "Help". The main area displays the following information:

Subsystem	Subsystem #1
Run Status	Completed
Last Run Time	Mar 4, 2019 9:49:50.000 AM
Number of Accounts	10000
Total Bytes Used	1,080,126,582
Number of Reads	1,079,235,759
Number of Writes	1,064,386,491
Bytes Read	0
Bytes Written	0

At the bottom of the window, there are two buttons: "Start Accounting" and "Dismiss".

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.

### **21.2.2.3. 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 will 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.

### 21.2.3. 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\_avaedit**). See the

**hpss\_avaedit** man page for more details.

- 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).

### 21.2.3.1. 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	d1k	3152	5A12x401	0
27	d1k	3152	5A12x501	0
341	d1k	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.

#### **21.2.3.2. 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.

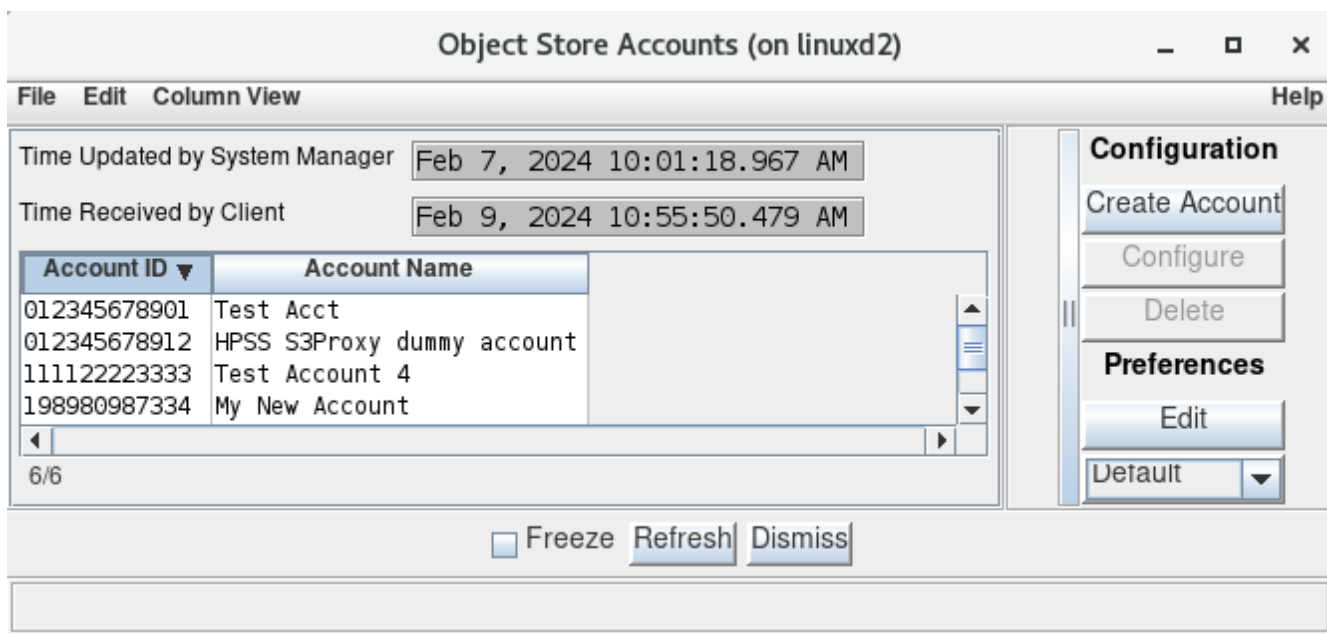
# Chapter 22. Object store accounts and accounting

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.

## 22.1. Managing object store accounts

### 22.1.1. 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.

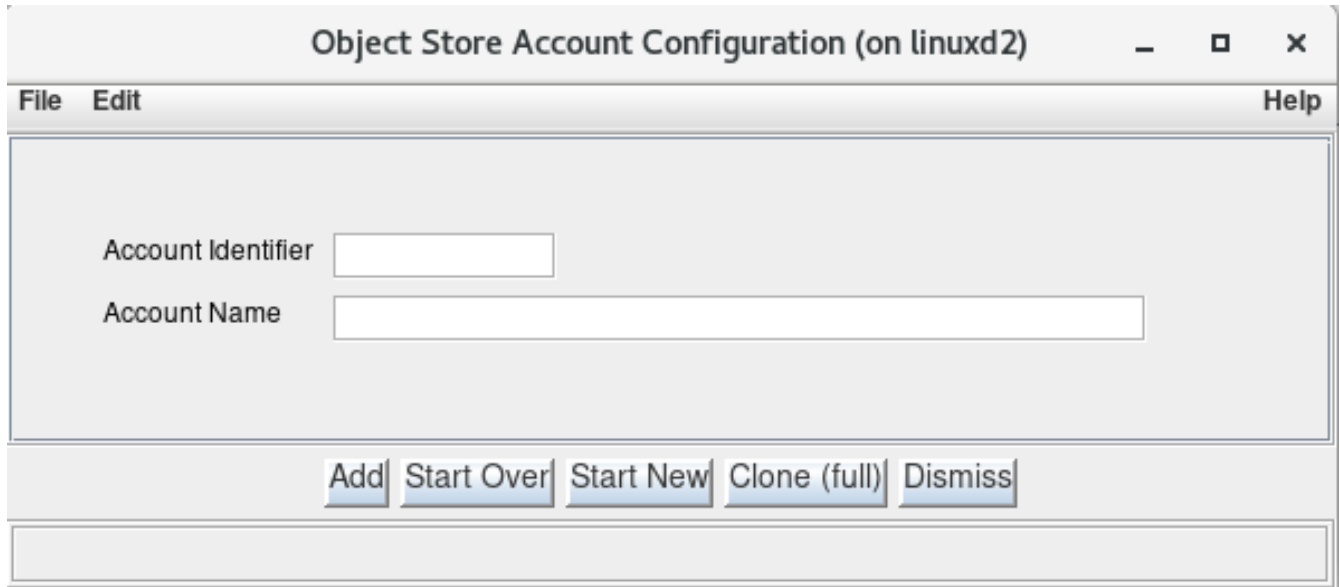
### Configure

Opens the selected object store account configurations for editing.

### Delete

Deletes the one or more selected object store accounts.

## 22.1.2. Object Store Account Configuration



The screenshot shows a window titled "Object Store Account Configuration (on linuxd2)". The window has a menu bar with "File", "Edit", and "Help". The main content area contains two input fields: "Account Identifier" and "Account Name". Below the input fields are five buttons: "Add", "Start Over", "Start New", "Clone (full)", and "Dismiss".

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.*

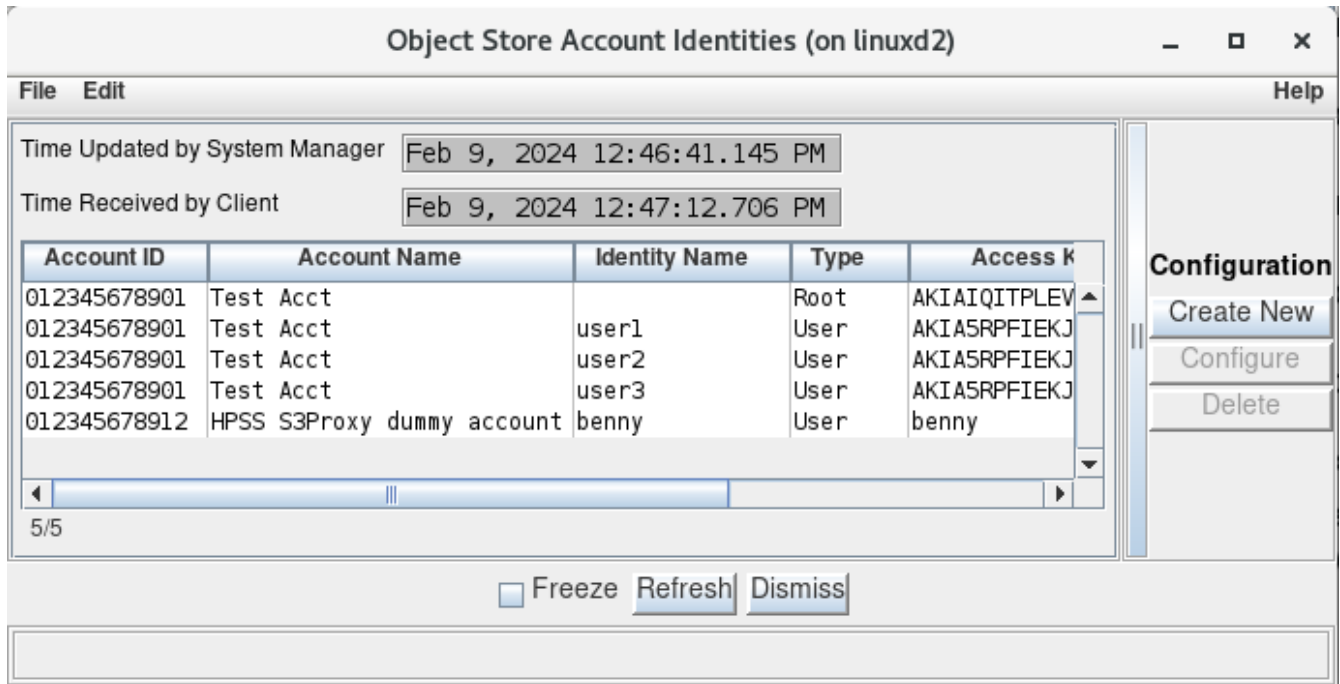
## 22.2. Managing object store account identities

Object store account identity configurations are used to authenticate HPSS when communicating

with the object store.

### 22.2.1. 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.

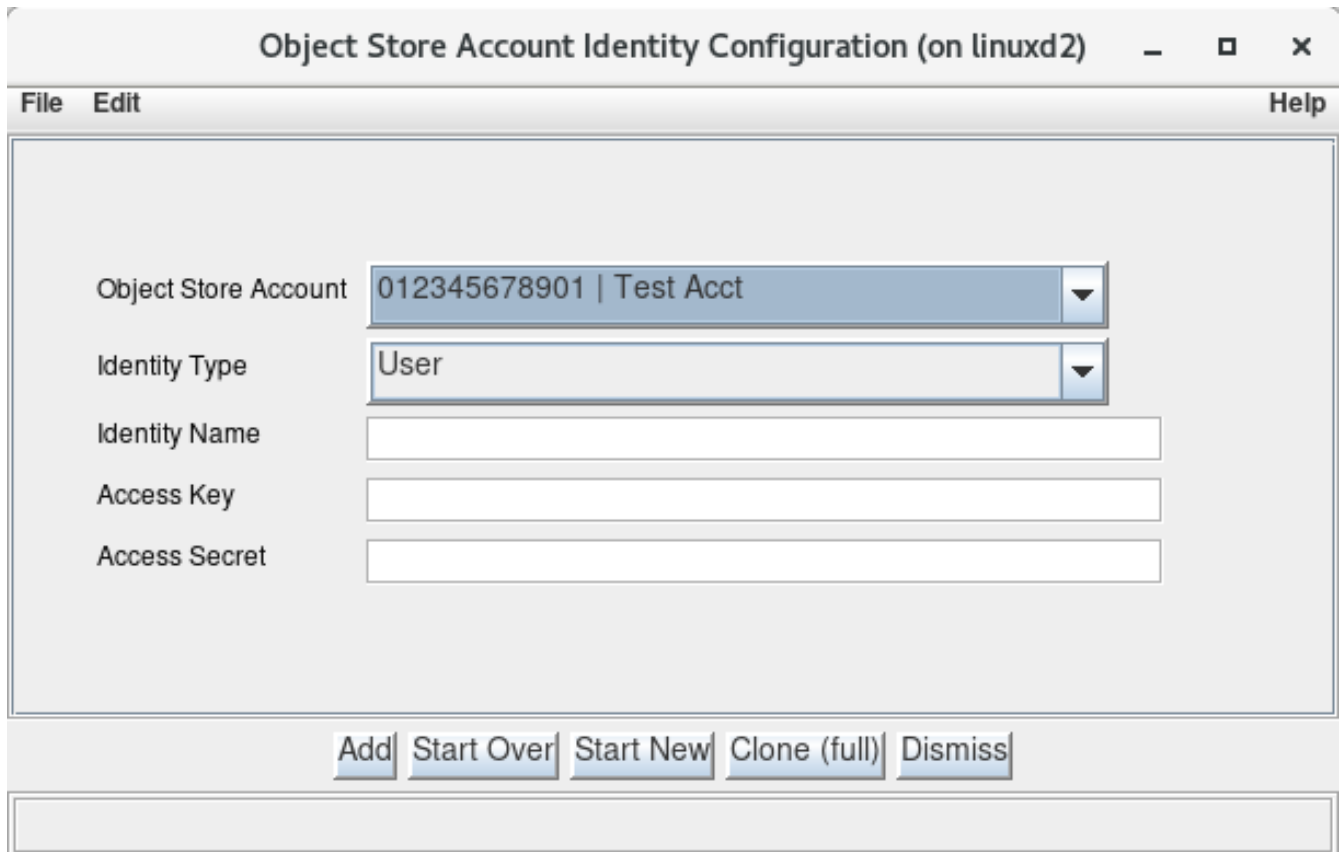
##### Configure

Opens the selected object store account identity configurations for editing.

##### Delete

Deletes the one or more selected object store account identities.

## 22.2.2. 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.*

## 22.3. Managing Object Store Account Buckets

The account bucket configuration associates object store buckets, regions, and optional key prefixes with an HPSS storage class.

### 22.3.1. Object Store Account Buckets window

An object store account bucket can be created and managed using the *Object Store Account Buckets* window.

The screenshot shows a window titled "Object Store Account Buckets (on linuxd2)". It has a menu bar with "File", "Edit", and "Help". Below the menu bar, there are two text fields: "Time Updated by System Manager" with the value "Jan 30, 2024 10:40:52.110 AM" and "Time Received by Client" with the value "Jan 31, 2024 11:15:14.167 AM".

Storage Class ID	Storage Class Name	Bucket Name	Object Name Prefix	Region
2	AWS Archive	ibm-hpss-backup		us-eas
2	AWS Archive	testbucket	myprefix	us-eas
2	AWS Archive	bucket		us-eas
2	AWS Archive	mybucket	account7/	us-eas
2	AWS Archive	ibm-hpss-2backup		us-eas
4	Storage Class 4	ibm-hpss-backup		us-eas
5	Storage Class 5	ibm-hpss-backup		us-eas
6	HPSS AWS/S3 Archive	mytestbucket		us-eas

At the bottom of the table, it says "8/8". To the right of the table is a "Configuration" sidebar with buttons for "Create New", "Configure", and "Delete". At the bottom of the window, there are buttons for "Freeze", "Refresh", and "Dismiss".

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.

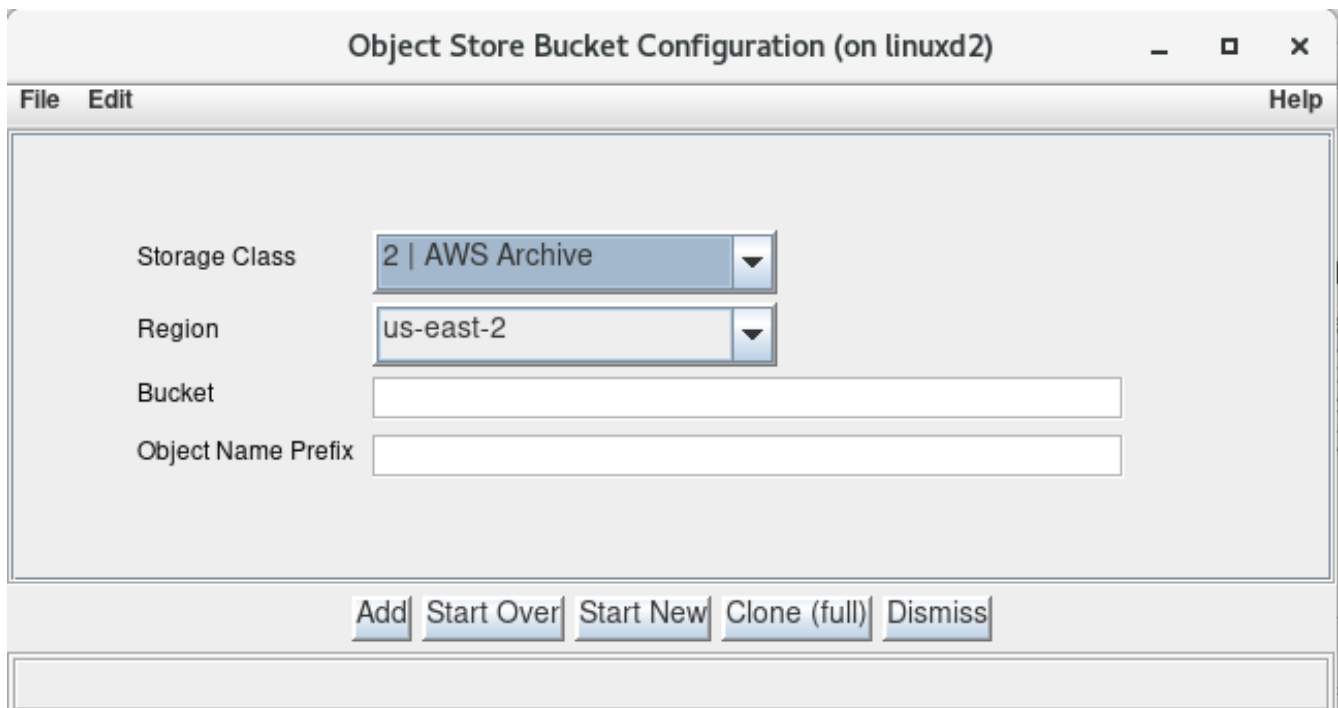
#### Configure

Opens the selected object store account bucket configurations for editing.

#### Delete

Deletes the one or more selected object store account buckets.

## 22.3.2. 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.



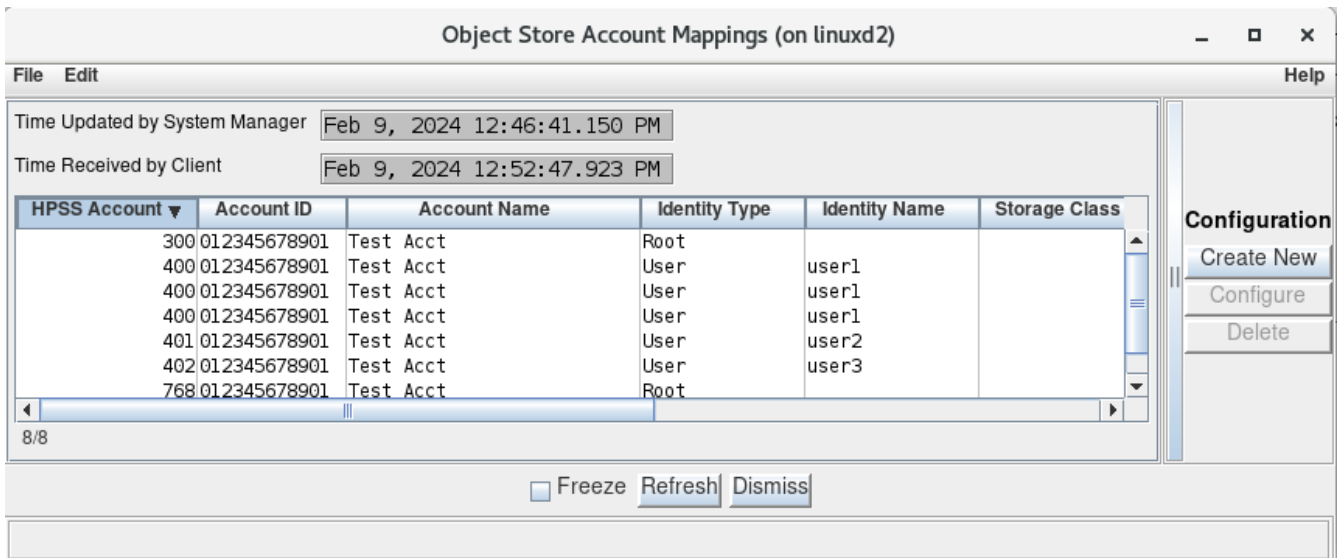
### Advice

This can be used to logically partition the contents of the bucket.

# 22.4. Managing Object Store Account Mappings

## 22.4.1. Object Store Account Mappings window

An object store account mapping can be created and managed using the *Object Store Account Mappings* window.



The screenshot shows a window titled "Object Store Account Mappings (on linuxd2)". It has a menu bar with "File", "Edit", and "Help". Below the menu bar, there are two text boxes: "Time Updated by System Manager" with the value "Feb 9, 2024 12:46:41.150 PM" and "Time Received by Client" with the value "Feb 9, 2024 12:52:47.923 PM". The main area contains a table with the following columns: "HPSS Account", "Account ID", "Account Name", "Identity Type", "Identity Name", and "Storage Class". The table has 8 rows of data. To the right of the table is a "Configuration" panel with buttons for "Create New", "Configure", and "Delete". At the bottom of the window, there are buttons for "Freeze", "Refresh", and "Dismiss".

HPSS Account	Account ID	Account Name	Identity Type	Identity Name	Storage Class
300	012345678901	Test Acct	Root		
400	012345678901	Test Acct	User	user1	
400	012345678901	Test Acct	User	user1	
400	012345678901	Test Acct	User	user1	
401	012345678901	Test Acct	User	user2	
402	012345678901	Test Acct	User	user3	
768	012345678901	Test Acct	Root		

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.

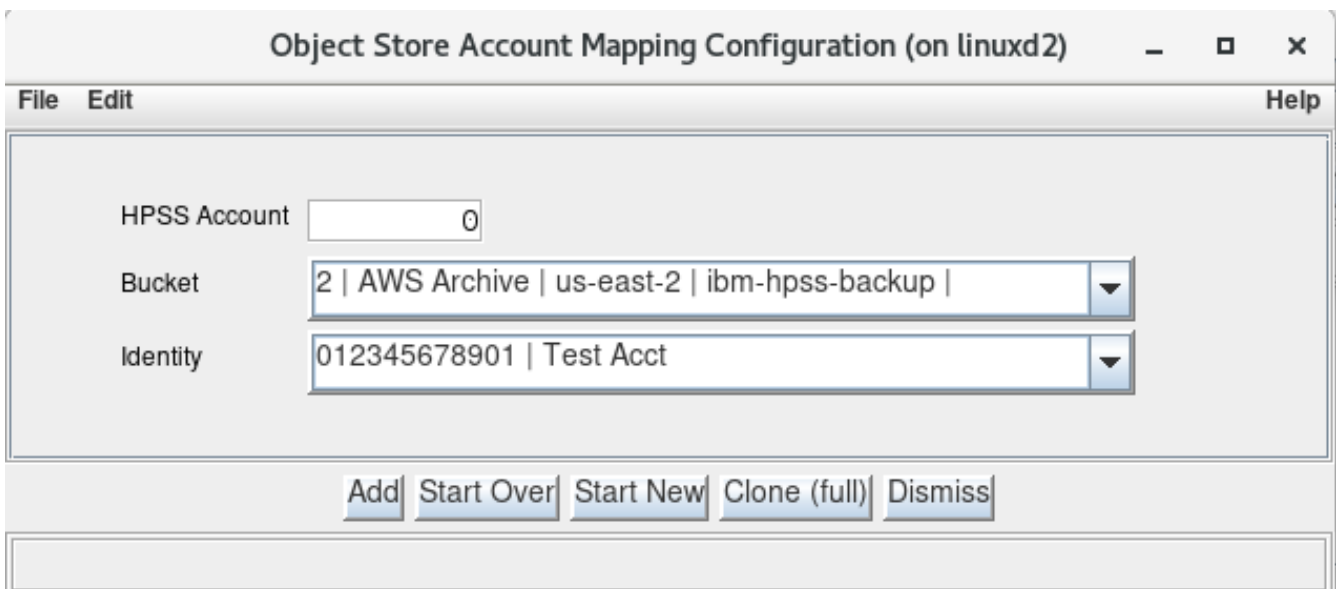
#### Configure

Opens the selected object store account mapping configurations for editing.

#### Delete

Deletes the one or more selected object store account mappings.

## 22.4.2. Object Store Account Mapping Configuration



The screenshot shows a window titled "Object Store Account Mapping Configuration (on linuxd2)". The window has a menu bar with "File", "Edit", and "Help". The main content area contains three input fields: "HPSS Account" with a text box containing "0", "Bucket" with a dropdown menu showing "2 | AWS Archive | us-east-2 | ibm-hpss-backup |", and "Identity" with a dropdown menu showing "012345678901 | Test Acct". Below these fields are five buttons: "Add", "Start Over", "Start New", "Clone (full)", and "Dismiss".

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 23. User interfaces

---

This chapter describes the configuration information for the user interfaces provided with HPSS for transferring files:

- Client Application Programming Interface (API)
- Parallel File Transfer Protocol (FTP) or PFTP

For other application configuration information, such as HPSSFS-FUSE, consult documentation bundled with the application.

## 23.1. 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 DMAP 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 that no there is 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\_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.

## **23.2. 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`, 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" binary to access HPSS the site *must* run the `hpss_pftpd` binary. The `hpss_pftpd` 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.tpl` and `hpssftp@.service.systemd.tpl` 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.tpl \
/etc/systemd/system/hpssftp.socket
cp /opt/hpss/config/templates/system_files/hpssftp@.service.systemd.tpl \
/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` man page 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.tpl` 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** 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** 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 `ftppaccess` 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 24. 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

## 24.1. 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.*

### 24.1.1. 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.

### **24.1.2. 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.

#### **24.1.2.1. 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.

### 24.1.3. 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.

## 24.2. 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.

### 24.2.1. 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

### 24.2.2. 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.

### 24.2.3. Kerberos backup

If Kerberos authentication is configured, periodic backups of the Kerberos infrastructure will need to be made. If `mkhps` 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

## 24.3. 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** or **recover** 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** or **recover** the suspect volume. See the **repack** and **recover** 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** utility. See the **recover** 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** utility. In some cases, **repack** 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** 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** 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** 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** by default aggregates the recovered files on the new volume. If for any reason the recovered files cannot be aggregated, **recover** 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** 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** 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** first to try to restore the data from an alternate copy.
  2. If you can't get all the data back from **recover**, run **repack** 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** 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** with the --**retry-failures** option, should be attempted first.
- If you don't have alternate copies:
  1. Run **repack** 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** 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** with the --**retry-failures** option, should be attempted first.

Once **recover** 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** is run for a volume and does not recover all of the files, and **repack** is then run and does finish retrieving all of the data, the recover operation must be cancelled. See the **recover** 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.

## 24.4. 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.

## 24.5. 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.

### 24.5.1. 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.

## 24.5.2. 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 25. Management tools

---

## 25.1. 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:

### 25.1.1. Fileset and junction management

These programs work for HPSS filesets.

#### **certjunction**

Creates an HPSS junction.

#### **deljunction**

Deletes an HPSS junction.

#### **lsfilesets**

Lists all HPSS filesets.

#### **lsjunctions**

Lists all HPSS junctions.

### 25.1.2. Tape management

These programs manage the life cycle of tapes in the system.

#### **repack**

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**

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**

Marks a disk or tape volume as read-only and not reclaimable.

#### **remove**

Deletes empty disk and tape volumes from the Core Server's metadata, then optionally exports the volumes from the PVL/PVR.

**recover**

Salvages data from damaged volumes by reading from other copies or other levels in the hierarchy.

**lbp\_verify**

Mounts tape volumes and verifies the data against stored CRCs.

These tools deal with shelving tapes:

**shelf\_tape**

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**

Performs various administrative functions related to shelved tapes.

### 25.1.3. System info

**dump\_sspvs**

Lists the physical volumes known to a particular subsystem, including statistics about file aggregates on tapes and RAIT volume status.

**dumpbf**

Lists the storage characteristics of a particular bitfile.

**lsvol**

Lists the path names of bitfiles that are located on a disk or tape volume.

**dumpppv\_pvl**

List the physical volumes managed by the PVL.

**dumpppv\_pvr**

List the physical volumes managed by a particular PVR.

**lshpss**

Lists various views of the system configuration.

**lsrb**

Lists a list of bitfiles that have not been accessed since a given date/time.

**lspvhist**

List physical—as opposed to virtual—tape volume history information.

**mps\_reporter**

Produces human-readable output from MPS summary files.

**plu**

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**

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**

Lists information all SCSI-attached devices including media-changers, disks, tape.

### **scsi\_home**

Displays SCSI home locations.

## **25.1.4. System management**

### **hpssadm**

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**

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**

Adds, manages, and removes HPSS users.

### **rtmu**

(Real-Time Monitoring Utility) Displays the state of requests in the HPSS system.

### **hpss\_avaedit**

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**

Translate HPSS error codes to descriptive strings.

### **hpss\_server\_acl**

Updates authorization data in the database.

### **hpss\_krb5\_keytab**

Generates a keytab in the form usable by the **hpssadm** program.



**hpss\_krb5\_rmxcred**

Removes expired cached Kerberos credentials from the default credential directory.

**hpss\_unix\_group**

Manages HPSS UNIX group file.

**hpss\_unix\_keygen**

Create an HPSS UNIX master key.

**hpss\_unix\_keytab**

A keytab generation program for the HPSS UNIX native security service.

**hpss\_unix\_passwd**

Changes the HPSS UNIX native security service password for an HPSS user.

**hpss\_unix\_user**

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**

Creates an XML index.

**xmldeleteindex**

Deletes an XML index.

**schema\_registrar**

Registers an XML schema with the database.

**hpsssum**

Checksums HPSS or local files

**25.1.5. User interfaces****pftp\_client**

Used to perform parallel file transfers between a local and remote host.

**25.1.6. Testing and debugging****scrub**

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.*

### 25.1.7. 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.

#### **hpsstats**

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.

## 25.2. 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.

### 25.2.1. 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
}
]
}

```

## 25.2.2. 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. + *Note*: 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.

## 25.3. 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*, which can be found at

<http://www-01.ibm.com/software/stormgmt/dfs/library/docs/31manuals/AdminGd/duagd008.htm>

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	<b>Execute (x)</b> on the directory itself. <b>Execute (x)</b> on all directories that lead to the directory.
List the contents of a directory	<b>Read (r)</b> on the directory itself. <b>Execute (x)</b> on all directories that lead to the directory.
List information about the objects in a directory	<b>Read (r)</b> and <b>execute (x)</b> on the directory itself. <b>Execute (x)</b> on all directories that lead to the directory.
Create an object	<b>Write (w)</b> , <b>execute (x)</b> on the directory in which the object is to be placed. <b>Execute (x)</b> on all directories that lead to the directory in which the object is to be placed.
Delete an object	<b>Write (w)</b> , <b>execute (x)</b> on the directory from which the object is to be deleted. <b>Execute (x)</b> on all directories that lead to the directory from which the object is to be deleted.
Rename an object	<p><b>Write (w)</b>, <b>execute (x)</b> on the object's current directory. <b>Execute (x)</b> on all directories that lead to the object's current directory.</p> <p><b>Write (w)</b>, <b>execute (x)</b> on the object's new directory. <b>Execute (x)</b> on all directories that lead to the object's new directory.</p> <p><b>Write (w)</b> on the object if the object is a directory.</p>
Read or read lock a file	<b>Read (r)</b> on the file itself. <b>Execute (x)</b> on all directories that lead to the file.
Write or write lock a file	<b>Write (w)</b> on the file itself. <b>Execute (x)</b> on all directories that lead to the file.
Execute a binary file	<b>Execute (x)</b> on the file itself. <b>Execute (x)</b> on all directories that lead to the file.

Execute a shell script	<b>Read (r)</b> and <b>execute (x)</b> on the script itself. <b>Execute (x)</b> on all directories that lead to the script.
List the ACLs on an object	<b>Execute (x)</b> on all directories that lead to the object.
Change the ACLs on an object	<b>Control (c)</b> on the object itself. <b>Execute (x)</b> on all directories that lead to the object.

# Appendix A: Glossary of terms and acronyms

---

<b>ACL</b>	Access Control List
<b>ACSLs</b>	Automated Cartridge System Library Software (Oracle StorageTek)
<b>ADIC</b>	Advanced Digital Information Corporation
<b>accounting</b>	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.
<b>ACI</b>	AML Client Interface
<b>AIX</b>	Advanced Interactive Executive. An operating system provided on many IBM machines.
<b>alarm</b>	A log record type used to report situations that require administrator investigation or intervention.
<b>AML</b>	Automated Media Library. A tape robot.
<b>AMS</b>	Archive Management Unit
<b>ANSI</b>	American National Standards Institute
<b>API</b>	Application Program Interface
<b>archive</b>	One or more interconnected storage systems of the same architecture.
<b>ASLR</b>	Address Space Layout Randomization
<b>attribute</b>	When referring to a managed object, an attribute is one discrete piece of information, or set of related information, within that object.

<b>attribute change</b>	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.
<b>audit (security)</b>	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.
<b>AV</b>	Account Validation
<b>bar code</b>	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).
<b>BFS</b>	HPSS Bitfile Service
<b>bitfile</b>	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.
<b>bitfile segment</b>	An internal metadata structure, not normally visible, used by the Core Server to map contiguous pieces of a bitfile to underlying storage.
<b>Bitfile Service</b>	Portion of the HPSS Core Server that provides a logical abstraction of bitfiles to its clients.
<b>BBTM</b>	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.
<b>CAP</b>	Cartridge Access Port
<b>cartridge</b>	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.
<b>class</b>	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.

<b>Class of Service</b>	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.
<b>cluster</b>	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.
<b>configuration</b>	The process of initializing or modifying various parameters affecting the behavior of an HPSS server or infrastructure service.
<b>COS</b>	Class of Service
<b>control path</b>	For the SCSI PVR, this is a connection to the library for sending commands. Control paths can be discovered using <code>device_scan</code> .
<b>Core Server</b>	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.
<b>CRC</b>	Cyclic Redundancy Check
<b>CS</b>	Core Server
<b>daemon</b>	A UNIX program that runs continuously in the background.
<b>DAS</b>	Distributed AML Server
<b>DB2</b>	A relational database system, a product of IBM Corporation, used by HPSS to store and manage HPSS system metadata.
<b>DCE</b>	Distributed Computing Environment
<b>debug</b>	A log record type used to report internal events that can be helpful in troubleshooting the system.

<b>delog</b>	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.
<b>deregistration</b>	The process of disabling notification to SSM for a particular attribute change.
<b>descriptive name</b>	A human-readable name for an HPSS server.
<b>device</b>	A physical piece of hardware, usually associated with a drive, that is capable of reading or writing data.
<b>directory</b>	An HPSS object that can contain files, symbolic links, hard links, and other directories.
<b>dismount</b>	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.
<b>DNS</b>	Domain Name Service
<b>DOE</b>	Department of Energy
<b>DPF</b>	Database Partitioning Feature
<b>drive</b>	A physical piece of hardware capable of reading or writing mounted cartridges. The terms device and drive are often used interchangeably.
<b>EB</b>	Exabyte ( $2^{60}$ )
<b>EOF</b>	End of File
<b>EOM</b>	End of Media
<b>ERA</b>	Extended Registry Attribute
<b>ESCON</b>	Enterprise System Connection
<b>event</b>	A log record type used to report informational messages (for example, subsystem starting or subsystem terminating).

<b>export</b>	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.
<b>FC SAN</b>	Fiber Channel Storage Area Network
<b>FIFO</b>	First in first out
<b>file</b>	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.
<b>file family</b>	An attribute of an HPSS file that is used to group a set of files on a common set of tape virtual volumes.
<b>fileset</b>	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.
<b>fileset ID</b>	A 64-bit number that uniquely identifies a fileset.
<b>fileset name</b>	A name that uniquely identifies a fileset.
<b>file system ID</b>	A 32-bit number that uniquely identifies an aggregate.
<b>FTP</b>	File Transfer Protocol
<b>FSF</b>	Forward Space File
<b>FSR</b>	Forward Space Record
<b>Gatekeeper</b>	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.
<b>Gatekeeping Service</b>	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.
<b>Gatekeeping Site Interface</b>	The APIs of the gatekeeping site policy code.

<b>Gatekeeping Site Policy</b>	The gatekeeping shared library code written by the site to monitor and throttle create, open, and/or stage requests.
<b>GB</b>	Gigabyte ( $2^{30}$ )
<b>GECOS</b>	The comment field in a UNIX password entry that can contain general information about a user, such as office or phone number.
<b>GID</b>	Group Identifier
<b>GK</b>	Gatekeeper
<b>GSS</b>	Generic Security Service
<b>GUI</b>	Graphical User Interface
<b>HA</b>	High Availability
<b>HACMP</b>	High Availability Clustered Multi-Processing - A software package used to implement high availability systems.
<b>HADR</b>	DB2 High Availability Disaster Recovery
<b>halt</b>	A forced shutdown of an HPSS server.
<b>HBA</b>	Host Bus Adapter
<b>HDM</b>	Shorthand for HPSS/DMAP.
<b>hierarchy</b>	See <i>storage hierarchy</i> .
<b>HPSS</b>	High Performance Storage System
<b>HPSS-only fileset</b>	An HPSS fileset that is not linked to an external file system (such as XFS).
<b>HTP</b>	HPSS Test Plan
<b>IBM</b>	International Business Machines Corporation
<b>ID</b>	Identifier
<b>IDE</b>	Integrated Drive Electronics



<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>import</b>	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.
<b>I/O</b>	Input/Output
<b>IOD/IOR</b>	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.
<b>IP</b>	Internet Protocol
<b>IRIX</b>	SGI's implementation of UNIX
<b>JRE</b>	Java Runtime Environment
<b>junction</b>	A mount point for an HPSS fileset.
<b>KB</b>	Kilobyte ( $2^{10}$ )
<b>KDC</b>	Key Distribution Center
<b>LAN</b>	Local Area Network
<b>LANL</b>	Los Alamos National Laboratory
<b>latency</b>	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.
<b>LBP</b>	Logical Block Protection
<b>LDAP</b>	Lightweight Directory Access Protocol
<b>LFT</b>	Local File Transfer
<b>LLNL</b>	Lawrence Livermore National Laboratory
<b>LMU</b>	Library Management Unit

<b>Location Service</b>	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.
<b>log record</b>	A message generated by an HPSS application and handled and recorded by the HPSS logging subsystem.
<b>log record type</b>	A log record may be of type alarm, event, info, debug, request, security, trace, or accounting.
<b>logging service</b>	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.
<b>LS</b>	Location Service
<b>LSM</b>	Library Storage Module
<b>LTO</b>	Linear Tape-Open. A half-inch open tape technology developed by IBM, HP, and Seagate.
<b>LUN</b>	Logical Unit Number
<b>LVM</b>	Logical Volume Manager
<b>MAC</b>	Mandatory Access Control
<b>managed object</b>	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.
<b>MB</b>	Megabyte ( $2^{20}$ )
<b>MBS</b>	Media Block Size
<b>metadata</b>	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.
<b>method</b>	A Java function or subroutine.

<b>migrate</b>	To copy file data from a level in the file's hierarchy onto the next lower level in the hierarchy.
<b>Migration/Purge Server</b>	An HPSS server responsible for supervising the placement of data in the storage hierarchies based upon site-defined migration and purge policies.
<b>MM</b>	Metadata Manager. A software library that provides a programming API to interface HPSS servers with the DB2 programming environment.
<b>mount</b>	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.
<b>mount point</b>	A place where a fileset is mounted in the XFS and HPSS name spaces.
<b>Mover</b>	An HPSS server that provides control of storage devices and data transfers within HPSS.
<b>MPS</b>	Migration/Purge Server
<b>MVR</b>	Mover
<b>NASA</b>	National Aeronautics and Space Administration
<b>Name Service</b>	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).
<b>name space</b>	The set of name-object pairs managed by the HPSS Core Server.
<b>NERSC</b>	National Energy Research Supercomputer Center
<b>NIS</b>	Network Information Service
<b>NLS</b>	National Language Support

<b>notification</b>	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.
<b>NS</b>	HPSS Name Service
<b>NSL</b>	National Storage Laboratory
<b>object</b>	See <i>managed object</i> .
<b>ORNL</b>	Oak Ridge National Laboratory
<b>OS</b>	Operating System
<b>OS/2</b>	The operating system (multi-tasking, single user) used on the AMU controller PC.
<b>PB</b>	Petabyte (2 <sup>50</sup> )
<b>PFTP</b>	Parallel File Transfer Protocol
<b>PFTPD</b>	PFTP Daemon
<b>physical volume</b>	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.
<b>Physical Volume Library</b>	An HPSS server that manages mounts and dismounts of HPSS physical volumes.
<b>Physical Volume Repository</b>	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.
<b>PIO</b>	Parallel I/O
<b>PIOFS</b>	Parallel I/O File System

<b>POSIX</b>	Portable Operating System Interface (for computer environments).
<b>purge</b>	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.
<b>purge lock</b>	A lock applied to a bitfile which prohibits the bitfile from being purged.
<b>PV</b>	Physical Volume
<b>PVL</b>	Physical Volume Library
<b>PVM</b>	Physical Volume Manager
<b>PVR</b>	Physical Volume Repository
<b>RAID</b>	Redundant Array of Independent Disks
<b>RAIT</b>	Redundant Array of Independent Tapes
<b>RAM</b>	Random Access Memory
<b>RAO</b>	Recommended Access Order
<b>reclaim</b>	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.
<b>registration</b>	The process by which SSM requests notification of changes to specified attributes of a managed object.
<b>reinitialization</b>	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.

<b>repack</b>	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.
<b>request</b>	A log record type used to report some action being performed by an HPSS server on behalf of a client.
<b>RISC</b>	Reduced Instruction Set Computer/Cycles
<b>RPC</b>	Remote Procedure Call
<b>RSF</b>	Reverse Space File
<b>RSR</b>	Reverse Space Record
<b>SCSI</b>	Small Computer Systems Interface
<b>security</b>	A log record type used to report security-related events (for example, authorization failures).
<b>SGI</b>	Silicon Graphics
<b>shelf tape</b>	A cartridge which has been physically removed from a tape library but whose file metadata still resides in HPSS.
<b>shutdown</b>	An HPSS SSM administrative operation that causes a server to stop its execution gracefully.
<b>sink</b>	The set of destinations to which data is sent during a data transfer, such as disk devices, memory buffers, or network addresses.
<b>SM</b>	System Manager
<b>SMC</b>	SCSI Medium Changer
<b>SME</b>	Subject Matter Expert
<b>SNL</b>	Sandia National Laboratories

<b>SOID</b>	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.
<b>source</b>	The set of origins from which data is received during a data transfer, such as disk devices, memory buffers, or network addresses.
<b>SP</b>	Scalable Processor
<b>SS</b>	HPSS Storage Service
<b>SSD</b>	Solid State Drive
<b>SSH</b>	Secure Shell
<b>SSI</b>	Storage Server Interface
<b>SSM</b>	Storage System Management
<b>SSM session</b>	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 <b>hpssgui</b> program, or it may be the command-line user interface provided by the <b>hpssadm</b> program.
<b>SSMSM</b>	Storage System Management System Manager
<b>stage</b>	To copy file data from a level in the file's hierarchy onto the top level in the hierarchy.
<b>start-up</b>	An HPSS SSM administrative operation that causes a server to begin execution.
<b>info</b>	A log record type used to report file staging and other kinds of information.
<b>STK</b>	Storage Technology Corporation (Oracle StorageTek)
<b>storage class</b>	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.

<b>storage hierarchy</b>	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.
<b>storage level</b>	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.
<b>storage map</b>	An HPSS object managed by the Core Server to keep track of allocated storage space.
<b>storage segment</b>	An HPSS object managed by the Core Server to provide abstract storage for a bitfile or parts of a bitfile.
<b>Storage Service</b>	The portion of the Core Server which provides control over a hierarchy of virtual and physical storage resources.
<b>storage subsystem</b>	A portion of the HPSS name space that is managed by an independent Core Server and (optionally) Migration/Purge Server.
<b>Storage System Management</b>	An HPSS component that provides monitoring and control of HPSS via a windowed operator interface or command-line interface.
<b>stripe length</b>	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).
<b>stripe width</b>	The number of physical volumes grouped together to represent a virtual volume.
<b>System Manager</b>	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 ( <b>hpssgui</b> ) and command line interface ( <b>hpssadm</b> ).



<b>TB</b>	Terabyte (2 <sup>40</sup> )
<b>TCP/IP</b>	Transmission Control Protocol/Internet Protocol
<b>TDS</b>	Tivoli Directory Server
<b>TI-RPC</b>	Transport-Independent-Remote Procedure Call
<b>trace</b>	A log record type used to record procedure entry/exit events during HPSS server software operation.
<b>transaction</b>	<p>A programming construct that enables multiple data operations to possess the following properties:</p> <ul style="list-style-type: none"> <li>• All operations commit or abort/roll-back together such that they form a single unit of work.</li> <li>• All data modified as part of the same transaction are guaranteed to maintain a consistent state whether the transaction is aborted or committed.</li> <li>• Data modified from one transaction are isolated from other transactions until the transaction is either committed or aborted.</li> <li>• Once the transaction commits, all changes to data are guaranteed to be permanent.</li> </ul>
<b>TSA/MP</b>	Tivoli System Automation for Multiplatforms
<b>TSM</b>	Tivoli Storage Manager
<b>UDA</b>	User-defined Attribute
<b>UDP</b>	User Datagram Protocol
<b>UID</b>	User Identifier
<b>UPC</b>	Universal Product Code
<b>UUID</b>	Universal Unique Identifier
<b>VPN</b>	Virtual Private Network

<b>virtual volume</b>	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).
<b>virtual volume block size</b>	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.
<b>VV</b>	Virtual Volume
<b>XDSM</b>	The Open Group's Data Storage Management standard. It defines APIs that use events to notify Data Management applications about operations on files.
<b>XFS</b>	A file system created by SGI available as open source for the Linux operating system.
<b>XML</b>	Extensible Markup Language

# 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 (ACSL) 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** configuration file contains tuning options to be used by HPSS clients and servers. For additional information, see the **HPSS.conf** manual page.

*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.tpl** 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.*

## 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)	<p>PFTP Client = \{</p> <p>Example: <code>PFTP Client = \{</code></p> <p><i>Optional Reserved</i> Stanza specifier</p> <p>Must be terminated with a matching "}"</p>
SubStanza	<p>SYSLOG Facility = &lt;value&gt;</p> <p>Values: <b>DAEMON, LOCAL0 ... LOCAL7</b></p> <p>Example: <code>SYSLOG Facility = LOCAL2</code></p> <p><i>Optional</i> SubStanza specifying the Syslog Facility for the Multinode Daemon</p>
SubStanza	<p>Debug Value = &lt;value&gt;</p> <p>Values: 0 - 3</p> <p>Example: <code>Debug Value = 1</code></p> <p><i>Optional</i> SubStanza specifying the Level of Debugging for the PFTP client. A larger number provides more information.</p>
SubStanza	<p>Protocol = &lt;value&gt;</p> <p>Values: <b>PDATA_AND_MOVER, PDATA_ONLY (default), PDATA_PUSH</b></p> <p>Example: <code>Protocol = PDATA_AND_MOVER</code></p> <p><i>Optional</i> SubStanza specifying the default protocol. May contain any of the three protocols supported.</p>
SubStanza	<p>Auto Parallel Size = &lt;value&gt;</p> <p>Value: Size specified as a decimal number or "xMB" style notation</p> <p>Example: <code>Auto Parallel Size = 4MB</code></p> <p><i>Optional</i> SubStanza specifying the minimum file size to start using the "auto-parallel" features of the PFTP client.</p>

Configuration type	Abbreviated description
SubStanza	<p>PortRange = &lt;value&gt;</p> <p>Value: <b>StartPort-EndPort</b></p> <p>Example: <b>PortRange = 10100-12100</b></p> <p><i>Optional</i> 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.</p> <p>Note: The old format (ncacn_ip_tcp[10100-12100]) is still supported for now, but will be phased out soon.</p>
SubStanza	<p>Parallel Block Size = &lt;value&gt;</p> <p>Value: Size specified as a decimal number or "xMB" style notation</p> <p>Example: <b>Parallel Block Size = 512KB</b></p> <p><i>Optional</i> SubStanza specifying the size of the data blocks to be used for parallel data transfers</p>
SubStanza	<p>Transfer Buffer Size = &lt;value&gt;</p> <p>Value: Size specified as a decimal number or "xMB" style notation</p> <p>Example: <b>Transfer Buffer Size = 16MB</b></p> <p><i>Optional</i> SubStanza specifying the PFTP client I/O buffer sizes</p>
SubStanza	<p>Socket Buffer Size = &lt;value&gt;</p> <p>Value: Viable socket size specified as a decimal number or "xMB" style notation</p> <p>Example: <b>Socket Buffer Size = 1MB</b></p> <p><i>Optional</i> SubStanza specifying the Pdata socket buffer sizes</p>



Configuration type	Abbreviated description
SubStanza	<p>MAX Ptran Size = &lt;value&gt;</p> <p>Value: Size specified as a decimal number or "xMB" style notation. NOLIMIT - Set to 250GB, the absolute max.</p> <p>Example: <code>MAX Ptran Size = 10GB</code></p> <p><i>Optional</i> SubStanza specifying a larger transfer size between socket open and closure. For disk COSSs, the segment sizes may potentially override this specification.</p>
SubStanza	<p>Enable SAN3P</p> <p>Example: <code>Enable SAN3P</code></p> <p><i>Optional</i> Enable SAN3P support. Default is SAN3P disabled. Note: this will also set the protocol to PDATA_AND_MOVER.</p>
SubStanza	<p>TA API Debug Level</p> <p>Values: 0 - 4</p> <p>Example: <code>TA API Debug Level = 0</code></p> <p><i>Optional</i> SubStanza specifying the level of debugging for the transfer agent. A larger number provides more information.</p>
SubStanza	<p>TA API Debug Log</p> <p>Value: Debug log file path</p> <p>Example: <code>TA API Debug Log = /var/TA/logs/PFTPC_TA_API_debug_%N_%P.log</code></p> <p><i>Optional</i> 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.</p>
SubStanza	<p>Special Features Enabled</p> <p>Example: <code>Special Features Enabled</code></p> <p><i>Optional</i> SubStanza for performance testing <i>only</i>. Should <i>not</i> be activated except by the appropriate personnel. Default is Off.</p>

Configuration type	Abbreviated description
SubStanza	<p>Disable stagebatch on mget</p> <p>Example: <code>Disable stagebatch on mget</code></p> <p><i>Optional</i> SubStanza specifying disabling using the batch stage site command to stage files prior to calling mget. Enabled unless this is active.</p>
SubStanza	<p>Use DirectIO</p> <p>Example: <code>Use DirectIO</code></p> <p><i>Optional</i> Enable Direct I/O for reads from the local file system. This is beneficial in cases where kernel caching may hinder performance.</p>
SubStanza	<p>DirectIO Block Size</p> <p>Example: <code>DirectIO Block Size = 512</code></p> <p><i>Optional</i> The block size of the local disk. Direct I/O requires that the transfer be block-aligned with the local device.</p>

Note: All PFTP Client SubStanzas are optional.

The **PFTP Client** = `{ ... }` Stanza contains several optional specifications for the **pftp\_client** 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 **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: `xMB`.

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: xMB.

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: xGB.

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 ncaen_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
}
```

## 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)	<p>PFTP Client Interfaces = \{</p> <p>Example: <code>PFTP Client Interfaces = \{</code></p> <p><i>Optional Reserved Stanza specifier</i></p> <p>Must be terminated with a matching "}"</p>
SubStanza (CMPD)	<p>&lt;Hostname&gt; &lt;hostname.domain&gt; = \{</p> <p>Example: <code>my_host my_host.my.domain = \{</code></p> <p>Contains the hostname(s) executing the PFTP client.</p> <p>Must be terminated with a matching "}"</p> <p>Multiple Hostname SubStanzas may be in a single <code>HPSS.conf</code> file representing multiple PFTP client hosts sharing the <code>HPSS.conf</code> file</p>
Section (CMPD)	<p>&lt;Name&gt; &lt;Name&gt; = \{</p> <p>Name: One or more hostnames</p> <p>Example: <code>storage storage.my.domain = \{</code></p> <p>Contains the hostname(s) executing the PFTP daemon.</p> <p>Must be terminated with a matching "}"</p> <p>Multiple Daemon Sections may be in a single Hostname SubStanza representing multiple PFTP daemon destinations which may use different characteristics</p>

Configuration type	Abbreviated description
SubSection	<p>&lt;Name&gt; or &lt;Dotted IP address&gt;</p> <p>Name: <b>Valid Interface Name</b></p> <p>Dotted IP address: <b>132.175.1.1</b></p> <p>Example: <b>eth0</b></p> <p>Example: <b>132.175.1.1</b></p> <p><i>Optional</i> parameter containing the name or dot notation IP address specification for the interface on the local host (<b>PFTP client</b>) to use to connect to the Mover(s) associated with the specified PFTP daemon</p>

The **PFTP Client Interfaces = { ... }** Stanza contains several configuration options for the **pftp\_client** 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
  ; }
; }
; }
```



# 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** 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)	<p>Multinode Table = \{</p> <p>Example: <code>Multinode Table = \{</code></p> <p><i>Optional Reserved</i> Stanza specifier</p> <p>Must be terminated with a matching "}"</p>
SubStanza	<p>Sleep for Debugger = values</p> <p>Value: Time in seconds</p> <p>Example: <code>Sleep for Debugger = 15</code></p> <p><i>Optional</i> parameter to specify a delay in the multinode daemons to allow diagnosis. This should <i>only</i> be specified for diagnostics and will unnecessarily cause degradation if misused. Leave commented out.</p>
SubStanza (CMPD)	<p>&lt;Local Hostname(s)&gt;</p> <p>Example: <code>my_host my_host.my.domain = \{</code></p> <p>Contains the local hostname(s) this SubStanza represents</p> <p>Must be terminated with a matching "}"</p>
Section (CMPD)	<p>&lt;Remote Hostname(s)&gt;</p> <p>Example: <code>FTP_host PFTP_host.domain = \{</code></p> <p>Contains the hostname(s) of the system running the PFTP server</p> <p>Must be terminated with a matching "}"</p>

Configuration type	Description
Sub-Section	<p data-bbox="363 210 564 241">&lt;remote_host&gt;</p> <p data-bbox="363 295 400 327">or</p> <p data-bbox="363 376 916 407">&lt;remote_host&gt; = &lt;dot notation interface&gt;</p> <p data-bbox="363 461 612 492">Example: <code>his_name</code></p> <p data-bbox="363 546 852 577">Example: <code>his_name = 100.102.103.45</code></p> <p data-bbox="363 631 1385 904">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.</p>

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**). 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
    }
  }
}
```

## 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.

$$BW * RTT \leq TCP / RTT$$

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)	<p>Network Options = \{</p> <p>Example: <b>Network Options = \{</b></p> <p><i>Optional Reserved Stanza specifier</i></p> <p>Must be terminated with a matching "}"</p>
SubStanza	<p>Default Write Size = &lt;value&gt;</p> <p>Example: <b>Default Receive = 1MB</b></p> <p><i>Optional SubStanza specifying the default network Read socket size if not specified explicitly</i></p>
SubStanza	<p>Default Write Size = &lt;value&gt;</p> <p>Example: <b>Default Send Size = 1MB</b></p> <p><i>Optional SubStanza specifying the default network write socket size if not specified explicitly</i></p>
SubStanza	<p>Default Write Size = &lt;value&gt;</p> <p>Example: <b>Default Write Size = 4MB</b></p> <p><i>Optional SubStanza specifying the default write size if not specified explicitly</i></p>

Configuration type	Description
SubStanza (CMPD)	<p data-bbox="363 210 794 241">&lt;Source IP Interface Name&gt; = \{</p> <p data-bbox="363 293 922 324">Example: <code>my_host my_host.my.domain = \{</code></p> <p data-bbox="363 376 1374 452"><i>Optional</i> SubStanza specifying an interface name. May contain one or more names separated by white spaces.</p> <p data-bbox="363 504 1358 580">May contain: "Default = \{" (<i>Reserved Specification</i>) for inclusion of entries not explicitly specified</p> <p data-bbox="363 631 895 663">Must be terminated with a matching "\}"</p>
Section (CMPD)	<p data-bbox="363 692 751 723">&lt;Destination IP address&gt; = \{</p> <p data-bbox="363 775 751 806">Example: <code>100.101.102.0 = \{</code></p> <p data-bbox="363 857 1358 934"><i>Optional</i> SubStanza specifying a dotted decimal address of the destination interface</p> <p data-bbox="363 985 1369 1061">Only one address is allowed; however, networks and sub-networks may be chosen by the appropriate specification of the NetMask.</p> <p data-bbox="363 1113 1358 1189">May contain: "Default = \{" (<i>Reserved Specification</i>) for inclusion of entries not explicitly specified</p> <p data-bbox="363 1240 895 1272">Must be terminated with a matching "\}"</p>
SubSection	<p data-bbox="363 1292 619 1323">NetMask = &lt;value&gt;</p> <p data-bbox="363 1375 847 1406">Value: Viable netmask as IP address</p> <p data-bbox="363 1458 826 1489">Example: <code>NetMask = 255.255.255.0</code></p> <p data-bbox="363 1541 1353 1617"><i>Optional</i> parameter to specify the dotted decimal net mask to apply to the destination IP address to determine whether the entry applies</p>
SubSection	<p data-bbox="363 1644 619 1675">RFC1323 = &lt;value&gt;</p> <p data-bbox="363 1727 512 1758">Values: 0, 1</p> <p data-bbox="363 1809 655 1841">Example: <code>RFC1323 = 1</code></p> <p data-bbox="363 1892 1289 1968"><i>Optional</i> parameter to specify whether the RFC1323 option should be disabled ("0") or enabled (any other value)</p>

Configuration type	Description
SubSection	<p>SendSpace = &lt;value&gt;</p> <p>Values: Size specified as decimal value or "xMB" format</p> <p>Example: <code>SendSpace = 256KB</code></p> <p><i>Optional</i> parameter to specify the value to be used for the socket sending buffer space</p>
SubSection	<p>RecvSpace = &lt;value&gt;</p> <p>Values: Size specified as decimal value or "xMB" format</p> <p>Example: <code>RecvSpace = 256KB</code></p> <p><i>Optional</i> parameter to specify the value to be used for the socket receive buffer space</p>
SubSection	<p>WriteSize = &lt;value&gt;</p> <p>Values: Size specified as decimal value or "xMB" format</p> <p>Example: <code>WriteSize = 1MB</code></p> <p><i>Optional</i> parameter used to set the size to be used for each individual write request to the network</p>
SubSection	<p>TcpNoDelay = &lt;value&gt;</p> <p>Values: 0, 1</p> <p>Example: <code>TcpNoDelay = 1</code></p> <p><i>Optional</i> parameter Indicates whether the TCP Delay option should be disabled (0) or enabled (any other value)</p>

**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. The maximum can be observed or changed on AIX using the **"no"** command and, respectively, observing or setting the **sb\_max** parameter. 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.



currently the ability to enable or disable **RFC 1323** support in this manner is specific to AIX. (An equivalent setting for Solaris is the **tcp\_wscale\_always** flag, set with the command "**ndd /dev/tcp tcp\_wscale\_always**".)

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 "=" and "{").



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
  }
}

```

## 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)	<p>PFTP Daemon = \{</p> <p><i>Optional</i> Reserved_Stanza specifier on client machines <i>only</i>. Required Stanza on all HPSS PFTP server systems.</p> <p>Must be terminated with a matching "}"</p>
SubStanza	<p>Allow Core Files</p> <p><i>Optional</i> SubStanza specifying that the system should save core files if the PFTP daemon crashes. By default, <b>xinetd</b> disables core files.</p>
SubStanza	<p>Core File Directory = &lt;value&gt;</p> <p>Value: <b>Pathname</b></p> <p>Example: <code>Core File Directory = /var/hpss/adm/core/PFTP_Daemon</code></p> <p><i>Optional</i> SubStanza to specify the directory where the PFTP server should put core files</p>
SubStanza	<p>TA API Debug Level</p> <p>Values: 0 - 4</p> <p>Example: <code>TA API Debug Level = 0</code></p> <p><i>Optional</i> SubStanza specifying the level of debugging for the transfer agent. A larger number provides more information.</p>

Configuration type	Abbreviated description
SubStanza	<p>TA API Debug Log</p> <p>Value: Debug log file path</p> <p>Example: <code>TA API Debug Log = /var/TA/logs/PFTPD_TA_API_debug_%N_%P.log</code></p> <p><i>Optional</i> SubStanza specifying the path to the transfer agent log file. A <b>%N</b> will be replaced by the number of agents, and a <b>%P</b> will be replaced by the PID of the particular agent.</p>
SubStanza	<p>SYSLOG Facility = &lt;value&gt;</p> <p>Value: DAEMON, LOCAL0 ... LOCAL7</p> <p>Example: <code>SYSLOG Facility = LOCAL0</code></p> <p>Replaces <b>-s&lt;string&gt;</b> option</p> <p><i>Optional</i> SubStanza specifying the syslog facility for the HPSS PFTPD. The default syslog facility is <b>DAEMON</b> (reference: <code>/usr/include/sys/syslog.h</code>). Alternatives are <b>LOCAL0 - LOCAL7</b>. Incorrect specification will default back to <b>DAEMON</b>. To make use of the alternates, modify <code>/etc/syslog.conf</code> to use the alternate facility. Note, the file specified in the <code>/etc/syslog.conf</code> must exist prior to initialization/refresh of the <b>syslogd</b>.</p>
SubStanza	<p>Use Foreign LDAP for Cross Realm</p> <p>Example: <code>Use Foreign LDAP for Cross Realm</code></p> <p><i>Optional</i> SubStanza specifying that LDAP lookups should use the LDAP server in the foreign realm. Few sites want this option.</p>
SubStanza	<p>Realms are Equivalent</p> <p>Value: off (default), on</p> <p>Example: <code>Realms are Equivalent = on</code></p> <p><i>Optional</i> SubStanza specifying that user in realm A is the entity as in realm B</p>

Configuration type	Abbreviated description
SubStanza	<p>FTP Base Directory = &lt;value&gt;</p> <p>Value: <b>Pathname</b></p> <p>Example: <code>FTP Base Directory = /var/hpss</code></p> <p>Replaces <b>-D&lt;string&gt;</b> option</p> <p><i>Optional</i> SubStanza setting the <b>{FTPBaseDir}</b> path. Default: <code>/var/hpss</code>. This directory must contain several subdirectories including: <code>adm</code>, <code>bin</code>, <code>daemon</code>, and <code>etc</code>. Specific files and subdirectories are located in each of these subdirectories - <code>etc</code>: <code>ftpaccess</code>, <code>[ftpbanner]</code>, and <code>ftpusers</code>. <code>adm</code>: <code>[daemon.syslog]</code>, <code>[hpss_ftpd.log]</code>, <code>[xferlog]</code>. <code>daemon</code>: <code>ftpd/ftp.pids-hpss_class</code>. <code>[ ]</code> implies optional others are required. <code>etc/passwd</code> is optional for FTP if <b>Use the KDC Registry</b> or <b>Use Extended Registry Attributes</b> is specified.</p>
SubStanza	<p>FTP Access File = &lt;value&gt;</p> <p>Value: <b>filename</b></p> <p>Example: <code>FTP Access File = myftpaccess</code></p> <p>Replaces <b>-F&lt;string&gt;</b> option</p> <p><i>Optional</i> SubStanza setting the <code>\{FTP_FtpAccessFile}</code>. Default: <code>ftpaccess</code>. Located in the directory <code>{FTPBaseDir}/etc</code>.</p>
SubStanza	<p>Disable Slash Home Directory</p> <p>Example: <code>Disable Slash Home Directory</code></p> <p>Replaces <b>-Z</b> option</p> <p><i>Optional</i> SubStanza disabling use of "/" (forward slash) as the user's home directory. Normally, this should be active for security reasons.</p>
SubStanza	<p>Disable Access if no Home Directory</p> <p>Example: <code>Disable Access if No Home Directory</code></p> <p>Replaces <b>-H</b> option</p> <p><i>Optional</i> 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.</p>

Configuration type	Abbreviated description
SubStanza	<p>HPSS FTP Principal = &lt;value&gt;</p> <p>Value: Appropriate HPSS Principal Name</p> <p>Example: <b>HPSS FTP Principal = hpssftp</b></p> <p>Replaces <b>-P</b> option and <b>hpss_option PRINC name</b> in <b>ftpaccess</b></p> <p><i>Optional</i> SubStanza specifying the HPSS principal representing <b>hpssftp</b> (HPSS only)</p>
SubStanza	<p>Disallow Passive Connections</p> <p>Example: <b>Disallow Passive Connections</b></p> <p><i>Optional</i> SubStanza disabling passive connections</p>
SubStanza	<p>Sleep for Debugger = &lt;value&gt;</p> <p>Example: <b>Sleep for Debugger = 5</b></p> <p>Replaces <b>-z</b> option</p> <p><i>Optional</i> 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.</p> <p><b>NOTE:</b> leaving this active will cause significant degradation to the PFTP service.</p>
SubStanza	<p>Must Have Credentials</p> <p>Example: <b>Must Have Credentials</b></p> <p>Replaces <b>-a</b> option</p> <p><i>Optional</i> SubStanza mandating authentication with Kerberos credentials, disabling {Username}/{Password} authentication. This also disables the <b>user</b> command.</p>
SubStanza	<p>Allow CCC Command</p> <p>Example: <b>Allow CCC Command</b></p> <p><i>Optional</i> Kerberos option not relevant to the HPSS PFTP daemon</p>

Configuration type	Abbreviated description
SubStanza	<p>PFTP IO Buffer Size = &lt;value&gt;</p> <p>Example: <code>PFTP IO Buffer Size = 4MB</code></p> <p>Replaces <b>-b&lt;string&gt;</b> option</p> <p><i>Optional</i> SubStanza setting the preferred IO Buffer Size for the PFTP server</p>
SubStanza	<p>Debug Value = &lt;value&gt;</p> <p>Example: <code>Debug Value = 3</code></p> <p>Replaces <b>-d</b> option(s)</p> <p><i>Optional</i> SubStanza specifying the level of debugging desired (1 - 4). Used internally to determine the quantity and detail of syslog messages from the PFTP daemon.</p>
SubStanza	<p>Non-Parallel HostName = &lt;value&gt;</p> <p>Example: <code>Non-Parallel HostName = aixrahe.sandia.gov</code></p> <p>Replaces <b>-h</b> option and <b>hpss_option HOSTname</b> in <code>ftpaccess</code></p> <p><i>Optional</i> SubStanza specifying the network interface to be used for data transferred between the PFTPD and the Movers when performing non-parallel transfers. Sets the <b>HPSS_API_HOSTNAME</b> environment variable for the Client API (HPSS only).</p>
SubStanza	<p>PFTP Debug Port = &lt;value&gt;</p> <p>Example: <code>PFTP Debug Port = 6666</code></p> <p>Replaces <b>-p&lt;port&gt;</b> option</p> <p><i>Optional</i> SubStanza specifying a port to be used by the HPSS PFTP daemon. Used only when initiating the daemon manually (rather than using <b>inetd/xinetd</b>). May be left on; it will not interfere with normal operations.</p>
SubStanza	<p>Default Time Out = &lt;value&gt;</p> <p>Example: <code>Default Time Out = 1500</code></p> <p>Replaces <b>-t</b> option and <b>hpss_option DTO time</b> in <code>ftpaccess</code></p> <p><i>Optional</i> SubStanza specifying the default timeout in seconds</p>

Configuration type	Abbreviated description
SubStanza	<p>Default Umask = &lt;value&gt;</p> <p>Example: <code>Default Umask = 077</code></p> <p>Replaces <b>-u</b> option and <b>hpss_option UMASK octal</b> in <code>ftpaccess</code></p> <p><i>Optional</i> SubStanza specifying the default umask in octal</p>
SubStanza	<p>Client API Verbose Value = &lt;value&gt;</p> <p>Example: <code>Client API Verbose Value = 1</code></p> <p>Replaces <b>-v</b> option(s)</p> <p><i>Optional</i> SubStanza specifying the level of HPSS Client API Logging to use (1 - 7). The Client API will perform logging specified by the <b>HPSS_DEBUG</b> environment variable in a file specified by the <b>HPSS_DEBUGPATH</b> environment variable.</p> <p>(Default name is <code>/var/hpss/ftp/adm/hpss_ftpd.log</code>.) The default value is <b>1</b> (HPSS only).</p>
SubStanza	<p>Disallow User Setting of COS</p> <p>Example: <code>Disallow User Setting of COS</code></p> <p>Replaces <b>-C</b> option</p> <p><i>Optional</i> SubStanza to disable the ability of clients to explicitly set the Class of Service for new files (via the "<b>site setcos</b>" command). Not recommended.</p>
SubStanza	<p>Maximum Time Out = &lt;value&gt;</p> <p>Value: Time in seconds</p> <p>Example: <code>Maximum Time Out = 86400</code></p> <p>Replaces <b>-T</b> option and <b>hpss_option MTO time</b> in <code>ftpaccess</code></p> <p><i>Optional</i> SubStanza specifying the maximum timeout in seconds</p>

Configuration type	Abbreviated description
SubStanza	<p>Use Extended Registry Attributes</p> <p>Example: <i>Use Extended Registry Attributes</i></p> <p>Replaces -X option</p> <p><i>Optional</i> SubStanza specifying use of the LDAP registry for authentication (bypassing the <b>passwd</b> file) and use of the <b>HPSS.homedir</b> and <b>HPSS.gecos</b> Extended Registry Attributes (ERAs) for the user's home directory and accounting fields (if they are filled)</p>
SubStanza	<p>Print Performance Data</p> <p>Example: <i>Print Performance Data</i></p> <p><i>Optional</i> SubStanza specifying the printing of additional performance numbers (non-HPSS PFTP daemon only)</p>
SubStanza	<p>Number of Ports = &lt;value&gt;</p> <p>Values: 1 - 64</p> <p>Example: <i>Number of Ports = 16</i></p> <p><i>Optional</i> SubStanza specifying the maximum stripe width allowed (non-HPSS PFTP daemon only)</p>
SubStanza	<p>PortRange = &lt;value&gt;</p> <p>Example: <i>PortRange = 10100-12100</i></p> <p><i>Optional</i> 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.</p>
SubStanza	<p>Socket Buffer Size = &lt;value&gt;</p> <p>Values: Viable Socket Sizes</p> <p>Example: <i>Socket Buffer Size = 1MB</i></p> <p><i>Optional</i> SubStanza specifying the socket buffer size (non-HPSS PFTP daemon only)</p>

Configuration type	Abbreviated description
SubStanza	<p>Set COS Based on Filesize</p> <p>Example: <code>Set COS Based on Filesize</code></p> <p><i>Optional</i> SubStanza specifying to set the COS from the FileSize Options table. Default: Ignore the COS in the table.</p>
SubStanza (Compound)	<p>FileSize Options = \{</p> <p>Example: <code>FileSize Options = \{</code></p> <p><i>Optional</i> SubStanza specifier</p> <p>Must be terminated with a matching <code>"}</code>"</p> <p>See notes below</p>
Section (Compound)	<p><code>&lt;value&gt; = \{</code></p> <p>Example: <code>1MB = \{</code></p> <p><i>Optional</i> Section specifier.</p> <p>Must be terminated with a matching <code>"}</code>"</p> <p>See notes below</p>
SubSection	<p>BlockSize = <code>&lt;value&gt;</code></p> <p>Example: <code>BlockSize = 512KB</code></p> <p><i>Optional</i> SubSection specifying the size of data blocks to be used based on file size. (Has no meaning for the HPSS PFTP daemon.)</p>
SubSection	<p>StripeWidth = <code>&lt;value&gt;</code></p> <p>Example: <code>StripeWidth = 0</code></p> <p><i>Optional</i> SubSection specifying the stripe width to be used based on file size. <b>0</b> (zero) means to use the Core Server Value (HPSS PFTP daemon) or use the default (non-HPSS PFTP daemon).</p>



Configuration type	Abbreviated description
SubSection	<p>COS = &lt;value&gt;</p> <p>Example: <code>COS = 2</code></p> <p><i>Optional</i> SubSection specifying the Class of Service to be used based on file size. <b>0</b> (zero) means to allow the Core Server to determine the optimal COS. (Has no meaning for the non-HPSS PFTD daemon.)</p>
SubStanza	<p>&lt;nodename&gt; Service Name = &lt;canonicalname&gt;</p> <p>Example: <code>sunrahe Service Name = sunrahe.sandia.gov</code></p> <p><i>Optional</i> 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 <code>gethostname()</code>. 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 <code>host/servicename@realm</code> or <code>ftp/servicename@realm</code> must exist in the Kerberos KDC and in the <code>/etc/v5srvtab</code> for the PFTP server executing on the machine <code>mymachine.ssm.com</code>. (Non-HPSS PFTP daemon only.)</p>
SubStanza	<p>Use System Password Files = &lt;value&gt;</p> <p>Example: <code>Use System Password Files = TRUE</code></p> <p>SubStanza specifying that the system password files (<code>/etc/passwd</code>, <code>/etc/group</code>, <code>/etc/shadow</code>) should be used. Should be specified explicitly. TRUE and FALSE are case-sensitive.</p>
SubStanza	<p>PFTP Password File = &lt;value&gt;</p> <p>Value: <b>Pathname/Filename</b></p> <p>Example: <code>PFTP Password File = /var/hpss/etc/passwd</code></p> <p><i>Optional</i> SubStanza used to specify the file containing the user's password information. <code>/var/hpss/etc/passwd</code> is the default.</p>
SubStanza	<p>PFTP Shadow File = &lt;value&gt;</p> <p>Value: <b>Pathname/Filename</b></p> <p>Example: <code>PFTP Shadow File = /var/hpss/etc/shadow</code></p> <p><i>Optional</i> SubStanza used to specify the file containing the user's protected password information. This should be specified if USERNAME/PASSWORD authentication is in effect.</p>

Configuration type	Abbreviated description
SubStanza	<p>PFTP Group File = &lt;value&gt;</p> <p>Value: <b>Pathname/Filename</b></p> <p>Example: <b>PFTP Group File = /var/hpss/etc/groups</b></p> <p><i>Optional</i> SubStanza used to specify the file containing the group information for PFTP clients. Default is <b>/var/hpss/etc/group</b>.</p>
SubStanza	<p>Primary Authentication Type = &lt;value&gt;</p> <p>Values: <b>krb5</b> (default), <b>spkm</b> (Not supported?), <b>unix</b></p> <p>Example: <b>Primary Authentication Type = krb5</b></p> <p><i>Optional</i> SubStanza used to specify the default authentication type</p>
SubStanza	<p>Primary Authenticator = &lt;value&gt;</p> <p>Values: &lt;auth_type&gt;:&lt;auth_file&gt; where &lt;auth_type&gt; = auth_keytab, auth_keyfile, auth_key, auth_password, auth_none</p> <p>Example: <b>Primary Authenticator = auth_keytab:/var/hpss/etc/hpss.keytab</b></p> <p><i>Optional</i> SubStanza used to specify the file containing the information to authenticate/authorize the <b>hpssftp</b> principal</p>
SubStanza	<p>Site Configuration File = &lt;value&gt;</p> <p>Values: <b>Pathname/Filename</b></p> <p>Example: <b>Site Configuration File = /var/hpss/etc/site.conf</b></p> <p><i>Optional</i> SubStanza used to specify the site configuration file to use with the Primary Authentication Mechanism</p>

Configuration type	Abbreviated description
SubStanza	<p>Secure Authentication = &lt;value&gt;</p> <p>Values: <b>none</b>, <b>required</b>, <b>required_same</b>, <b>accepted</b></p> <p><i>Optional</i> SubStanza used to specify the requirements for secure authentication.</p> <ul style="list-style-type: none"> <li>• <b>none</b> means it is not allowed</li> <li>• <b>required</b> means it is required, but the login provided in the USER command does not need to match.</li> <li>• <b>required_same</b> means that the USER command must match the authenticated name.</li> <li>• <b>accepted</b> means secure authentication is accepted but does not affect the USER/PASS cases.</li> </ul> <p>When the USER command does match the securely authenticated principal, the PASS command will be optional and ignored if given.</p>
SubStanza	<p>Mapfile Specifier = &lt;value&gt;</p> <p>Values: <b>Pathname/filename</b></p> <p>Example: <b>Mapfile Specifier = /var/hpss/etc/MapfileName</b></p> <p><i>Optional</i> 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 <i>must</i> be protected for security reasons. These files should be owned by <i>root</i> and readable and writable by <i>root</i> only.</p>
SubSection	<p>Default Authorization Mechanism = &lt;value&gt;</p> <p>Values: <b>LDAP</b>, <b>UNIX</b>, <b>DBAS</b> (not implemented)</p> <p>Example: <b>Default Authorization Mechanism = LDAP</b></p> <p><i>Optional</i> 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. <b>DBAS</b> has <i>not</i> been implemented in PFTP or in HPSS.</p>

Configuration type	Abbreviated description
SubSection	<p>Use Site Auth Method = &lt;value&gt;</p> <p>Values: CryptoCard, KRB5KDC, SecurID, NIST</p> <p>Example: <code>Use Site Auth Method = CRYPTOCARD</code></p> <p><i>Optional</i> 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 hpsd_pftpd with site-specific modules linked in. <b>NOTE:</b> if this option is specified, the UserName/Password Mechanism will <i>not</i> use a standard password.</p>
SubStanza	<p>{Hostname} Service Name = {servicename}</p> <p>Example: <code>mymachine Service Name = fire.clearlake.ibm.com</code></p> <p><i>Optional</i> 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.</p> <p>This is very useful for computing clusters and multi-homed systems using a Kerberized PFTP server.</p>

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/PFTP_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 = aixrahe.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 multiple "machinename Service Name" entries if this
# file is common to multiple PFTP Daemon servers.
; aixrahe.sandia.gov Service Name = aixrahe.sandia.gov
; sunrahe Service Name = sunrahe.sandia.gov
}

```

## 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)	<p>Transfer Agent = \{</p> <p><i>Reserved</i> Stanza specifier</p> <p>Must be terminated with a matching "}"</p>
SubStanza	<p>Agent Debug Level = &lt;value&gt;</p> <p>Value: 0 - 4</p> <p>Example: <code>Agent Debug Level = 1</code></p> <p><i>Optional</i> Sets the debug level (where 0=no debug, 4=max debug) for transfer agent processes</p>
SubStanza	<p>Debug Log File = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <code>Debug Logfile = /var/TA/logs/agent_debug_%N_%P.log</code></p> <p><i>Optional</i> Specifies the debugging output file. It can be overridden by environment variable settings.</p>
SubStanza	<p>Enable Core Dump = &lt;value&gt;</p> <p>Value: <b>YES/NO</b></p> <p>Example: <code>Enable Core Dump = YES</code></p> <p><i>Optional</i> Determines whether the agent will enable full core dumps</p>
SubStanza	<p>Working Directory = &lt;value&gt;</p> <p>Value: <b>Pathname</b></p> <p>Example: <code>Working Directory = /var/TA/cores</code></p> <p><i>Optional</i> 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.</p>

Configuration type	Abbreviated description
SubStanza	<p>Nodeset File = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <b>Nodeset File = /usr/local/etc/nodeset.conf</b></p> <p><i>Optional</i> File containing named sets of nodes that can be referred to via the "SET:setname" notation</p>
SubStanza	<p>Node Affinity File = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <b>Node Affinity File = /usr/local/etc/node_affinity.conf</b></p> <p><i>Optional</i> File used to specify groups of nodes that are able to communicate in a network whose topology does not support full interconnection</p>
SubStanza	<p>Shared FS File = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <b>Shared FS File = /usr/local/etc/shared_fs.conf</b></p> <p><i>Optional</i> File used to specify the list of shared file system mount points and associated nodes or NodeSets</p>
SubStanza	<p>Agent File = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <b>Agent File = /usr/local/etc/agent.conf</b></p> <p><i>Optional</i> File containing the list of client hosts and the list of nodes that can be used as agents from each client</p>
SubStanza	<p>Disabled Node File = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <b>Disabled Node File = /usr/local/etc/disabled_node.conf</b></p> <p><i>Optional</i> File used to disable selection of nodes that are temporarily unavailable. It overrides entries in the Agent File.</p>

Configuration type	Abbreviated description
SubStanza	<p>Audit Logfile = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <i>Audit Logfile = /var/hpss/log/PMTA_AuditLog</i></p> <p><i>Optional</i> File used to specify the location of the transfer agent audit log</p>
SubStanza	<p>Debug Logfile = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <i>Debug Logfile = /var/TA/log/pmta_Audit_%N_%P.log</i></p> <p><i>Optional</i> Contains an audit trail of all PMTA activity</p>
SubStanza	<p>Agent Auth Mechanism = &lt;value&gt;</p> <p>Value: <b>none (default), ident, kerberos</b></p> <p>Example: <i>Agent Auth Mechanism = none</i></p> <p><i>Optional</i> The authentication mechanism used by the agent to verify its parent's identity</p>
SubStanza	<p>Agent Authenticator Type = &lt;value&gt;</p> <p>Value: <b>none (ident), token (future), kerberos (future)</b></p> <p>Example: <i>Agent Authenticator Type = none</i></p> <p><i>Optional</i> The mechanism-specific type of authenticator used for verifying the parent's identity</p>
SubStanza	<p>Agent Authenticator File = &lt;value&gt;</p> <p>Value: <b>Pathname/Filename</b></p> <p>Example: <i>Agent Authenticator File = /usr/local/etc/ident_hosts.conf</i></p> <p><i>Optional</i> 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</p>

Configuration type	Abbreviated description
SubStanza	<p>SYSLOG Facility = &lt;value&gt;</p> <p>Values: <b>off, LOCAL0 ... LOCAL7</b></p> <p>Example: <code>SYSLOG Facility = LOCAL0</code></p> <p><i>Optional</i> Controls logging to syslog</p>
SubStanza	<p>Allow Uid Mapping = &lt;value&gt;</p> <p>Values: <b>YES, NO (default)</b></p> <p>Example: <code>Allow Uid Mapping = YES</code></p> <p><i>Optional</i> Specifies whether the same user can have different UIDs on different machines within the site</p>
SubStanza	<p>Uid Mapfile = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <code>Uid Mapfile = /usr/local/etc/uid_mapfile</code></p> <p><i>Optional</i> The name of the mapping file when "Allow Uid Mapping" is set to "YES"</p>
SubStanza	<p>Allow Gid Mapping = &lt;value&gt;</p> <p>Values: <b>YES, NO (default)</b></p> <p>Example: <code>Allow Gid Mapping = YES</code></p> <p><i>Optional</i> Specifies whether the same group can have different GIDs on different machines within the site</p>
SubStanza	<p>Gid Mapfile = &lt;value&gt;</p> <p>Value: <b>Pathname/filename</b></p> <p>Example: <code>Gid Mapfile = /usr/local/etc/gid_mapfile</code></p> <p><i>Optional</i> The name of the mapping file when "Allow Gid Mapping" is set to "YES"</p>

Configuration type	Abbreviated description
SubStanza	<p>Umask = &lt;value&gt;</p> <p>Value: <b>Octal 000 - 777</b></p> <p>Example: <b>Umask = 002</b></p> <p><i>Optional</i> Umask value that the transfer agent should set when it starts up</p>
SubStanza	<p>Connections per NIC = &lt;value&gt;</p> <p>Value: <b>1 (default) - 8</b></p> <p>Example: <b>Connections per NIC = 4</b></p> <p><i>Optional</i> 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.</p>
SubStanza	<p>Max Local IO Count = &lt;value&gt;</p> <p>Value: <b>1 - 8 (default = 4)</b></p> <p>Example: <b>Max Local IO Count = 4</b></p> <p><i>Optional</i> The maximum number of concurrent local file I/O ops</p>
SubStanza	<p>File Create Mode = &lt;value&gt;</p> <p>Value: <b>Octal 000 - 777</b></p> <p>Example: <b>File Create Mode = 640</b></p> <p><i>Optional</i> The permissions that should be specified when the transfer agent creates files</p>
SubStanza	<p>Stripe Size = &lt;value&gt;</p> <p>Value: <b>128KB - 256MB</b></p> <p>Example: <b>Stripe Size = 16MB</b></p> <p><i>Optional</i> 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.</p>

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

```

## 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



# HPSS Environment Variables

## 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.

## 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

## 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

## 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 DMAPAPI 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 ("hpsssm")**

Principal name for SSM

**HPSS\_PRINCIPAL\_ADM\_USER ("\${HPSS\_USER}")**

Principal name for primary administrator

## 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\_RAID\_UID ("311")**

UID for RAID

**HPSS\_PRINCIPAL\_SSM\_UID ("312")**

UID for SSM

## 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

## 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

## 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

# Accounting 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

# 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

# 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

# Database environment variables

Environment variables related to database

**HPSS\_DB\_INSTANCE\_OWNER** ("hpsbdb")

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+

## 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 descriptive 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

## 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 ("2000")**

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\_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

# 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 DMAP 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

## **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\_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.

## 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\_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\_DISCOVERY\_THREADS ("20")**

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 ("20")**

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\_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.

## 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

## 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

## 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

## 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

# 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

# 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.

# 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.

## 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

## 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\_SOCKET\_SNDBUF\_SZ (NULL)**

The RPC socket send buffer size

**HPSS\_RPC\_SOCKET\_RCVBUF\_SZ (NULL)**

The RPC socket receive buffer size

**HPSS\_RPC\_SOCKET\_IO\_SZ (NULL)**

The RPC socket I/O size to be used

**HPSS\_RPC\_SOCKET\_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

## 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

## 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.

## 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

## HPSS Group Names

HPSS groups

**HPSS\_GROUP** ("**`{HPSS_GRP_NAME}`**")

HPSS group name

**HPSS\_GROUP\_DB ("hpsddb")**

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

## 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

## 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

## 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

## 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

## HPSS Object Store Notification 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 `mkhps` utility or hand-created. Be very careful when using `mkhps` 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.tpl`. See [Appendix D: HPSS.conf configuration file](#) for more details.
- **authz.conf.** Created by `mkhps`. 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 `mkhps`. 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 `mkhps`. 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 **mkhps**. 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** utility to add each of the HPSS server principals to the file. See the HPSS man page for the usage of **hpss\_unix\_keytab**.
- **mm.keytab**. Created by **mkhps**. 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**. Created by **mkhps**. Contains the site name, realm name, realm ID, authorization mechanism and authorization URL to utilize. 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 **mkhps**. 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 **mkhps**. 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 **mkhps**. 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 **mkhps**. 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





- An optional subdirectory to hold keytabs for **hpssadm** 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** and **hpssadm**). 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.