

---

# ***HPSS***

## ***User's Guide***

**High Performance Storage System**  
**Release 4.3**

*August 2001*

---

## HPSS User's Guide

© 1992-2000 International Business Machines Corporation, The Regents of the University of California, Sandia Corporation, Lockheed Martin Energy Research Corporation, and NASA Langley Research Center.

All rights reserved.

Portions of this work were produced by the University of California, Lawrence Livermore National Laboratory (LLNL) under Contract No. W-7405-ENG-48 with the U.S. Department of Energy (DOE), by the University of California, Lawrence Berkeley National Laboratory (LBNL) under Contract No. DEAC03776SF00098 with DOE, by the University of California, Los Alamos National Laboratory (LANL) under Contract No. W-7405-ENG-36 with DOE, by Sandia Corporation, Sandia National Laboratories (SNL) under Contract No. DEAC0494AL85000 with DOE, and Lockheed Martin Energy Research Corporation, Oak Ridge National Laboratory (ORNL) under Contract No. DE-AC05-96OR22464 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, Sandia Corporation, Lockheed Martin Energy Research Corporation, 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.

Printed in the United States of America.

HPSS Release 4.3

August 2001

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

AIX and RISC/6000 are trademarks of International Business Machines Corporation.

Encina is a registered trademark of Transarc Corporation.

UNIX is a registered trademark of Unix System Laboratories, Inc.

Sammi is a trademark of Scientific Software Intercomp.

NFS and Network File System are trademarks of Sun Microsystems, Inc.

DST is a trademark of Ampex Systems Corporation.

ACLS is a trademark of Storage Technology Corporation. Other brands and product names appearing herein may be trademarks or registered trademarks of third parties.

---

## Preface

This High Performance Storage System (HPSS) User's Guide provides the necessary information for transferring files using HPSS. It is designed for HPSS users.

The document provides the User's Guide information for HPSS Release 4.2. In particular, the following interfaces are described: standard File Transfer Protocol (FTP) interface, Parallel FTP (PFTP) interface, Network File System Version 2 (NFS V2), and Distributed File System (DFS). **Note:** It is not the intent of this document to define the standard commands and subcommands provided by standard FTP, NFS, and DFS. Only interface extensions provided by HPSS are defined within the HPSS User's Guide.

Refer to the HPSS Installation Guide and HPSS Management Guide for descriptions of the interfaces provided to HPSS administrators. Programming interfaces are documented in the HPSS Programmer's Reference Guides.

Refer to the HPSS Programmer's Reference Guide, Volume 1 for programming interfaces provided to the end user. The programming interfaces provided in the Programmer's Guide Volume 1 are: HPSS Client Application Programming Interface (API), and 64 bit arithmetic functions.

Refer to the HPSS Error Messages Manual for a list of all HPSS error and advisory messages which are output by the HPSS software. For each message, the following information is provided: message identifier and text, source file name(s) which generated the message, problem description, system action, and administrator action.

Refer to the HPSS Programmer's Reference Guide, Volume 2 for programming interfaces to each HPSS server. While it is envisioned that most users will access HPSS through the client API, standard FTP, PFTP, NFS, or the DFS interfaces (documented in Volume 1), well defined programming interfaces are defined for each HPSS server. It should be noted that programming to the individual server level is a more complex programming model which requires a greater understanding of the HPSS servers.

The HPSS User's Guide is structured as follows:

*Chapter 1: Overview* This chapter provides an overview of each type of user interface, a summary of key storage concepts, and recommendations on usage.

*Chapter 2: File Transfer Protocol (FTP)* This chapter defines the extensions to the standard FTP interface.

*Chapter 3: Parallel File Transfer Protocol (PFTP)* This chapter defines the Parallel FTP (PFTP) interface.

*Chapter 4: User Utilities* This chapter defines the set of utilities available to the general user.

*Appendix A: Acronyms* This appendix A provides a list of acronyms used in this document.

*Appendix B: References* This appendix lists documents cited in the text as well as other reference materials.

---

## Typographic and Keying Conventions

This document uses the following typographic conventions:

**Bold Bold** words or characters represent system elements that you must use literally, such as functions, commands or keywords.

*Italic Italic* words or characters represent variable values to be supplied.

[ ] Brackets enclose optional items in syntax and format descriptions.

{ } Braces enclose a list of items to select in syntax and format descriptions.

---

# Table of Contents

<b>Preface</b> .....	<b>5</b>
<b>Chapter 1. Overview</b> .....	<b>1-1</b>
1.1. User Interfaces .....	1-1
1.1.1. File Transfer Protocol (FTP) .....	1-1
1.1.2. Parallel FTP (PFTP).....	1-1
1.1.3. Network File System Version 3 (NFS) .....	1-2
1.1.4. Distributed File System (DFS).....	1-3
1.1.5. User Utilities .....	1-4
1.2. Storage Concepts.....	1-5
1.2.1. Class of Service.....	1-5
1.2.2. Storage Class.....	1-6
1.2.3. Storage Hierarchy .....	1-6
1.3. Interface Usage Considerations .....	1-7
1.4. User IDs.....	1-8
1.5. DCE User Accounts .....	1-9
<b>Chapter 2. File Transfer Protocol (FTP)</b> .....	<b>2-1</b>
2.1. Site Commands.....	2-1
2.1.1. Specifying a File's Class of Service - setcos .....	2-2
2.1.2. Changing a File's Group by ID - chgid .....	2-2
2.1.3. Changing a File's Group by Name - chgrp .....	2-2
2.1.4. Changing a File's Permissions - chmod .....	2-3
2.1.5. Changing a File's Owner by Name - chown .....	2-4
2.1.6. Changing a File's Owner by ID - chuid .....	2-4
2.1.7. Staging a File - stage .....	2-5
2.1.8. Setting the Desire Wait Options (for Migrated Files) - wait .....	2-5
2.1.9. Creating a Symbolic Link - symlink.....	2-6
2.1.10. Allocating space for files - quote allo64.....	2-6
2.2. List Directory Extensions .....	2-7
<b>Chapter 3. Parallel File Transfer Protocol (PFTP)</b> .....	<b>3-1</b>
3.1. <b>Parallel FTP Client Transfers</b> .....	<b>3-3</b>
3.1.1. <b>Parallel FTP Client Configuration</b> .....	<b>3-4</b>
3.2. <b>Additional HPSS Commands</b> .....	<b>3-6</b>
3.2.1. General Login messages (Examples) .....	3-7
3.2.2. <b>Parallel append - pappend</b> .....	<b>3-8</b>
3.2.3. <b>Parallel file store - pput</b> .....	<b>3-10</b>
3.2.4. <b>Parallel file store - mpput</b> .....	<b>3-12</b>
3.2.5. <b>Parallel file retrieval - pget</b> .....	<b>3-14</b>
3.2.6. <b>Parallel file retrieval - mpget</b> .....	<b>3-16</b>
3.2.7. <b>Local File append - lfappend</b> .....	<b>3-18</b>
3.2.8. <b>Local File store - lfput</b> .....	<b>3-20</b>
3.2.9. <b>Local file retrieval - lfget</b> .....	<b>3-22</b>
3.2.10. <b>Multiple Local file store - mlfput</b> .....	<b>3-24</b>

---

3.2.11. Multiple Local file retrieval - mlfget .....	3-26
3.2.12. Specify TCP socket based transfers - psocket .....	3-28
3.2.13. Specify transfer stripe width - setpwidth .....	3-29
3.2.14. Specify transfer block size - setpblocksize .....	3-31
3.2.15. Multinode Enable/Disable - multinode .....	3-33
3.2.16. Autoparallel Enable/Disable - autoparallel.....	3-34
3.2.17. Get Current Protocol Mode - getprot.....	3-35
3.2.18. 3.2.19. Get Tuning Parameters - gettuningparms .....	3-36
3.2.19. Set the PDATA_ONLY protocol - pdata .....	3-38
3.2.20. Override the Non-AutoParallel mode - penable.....	3-39
3.2.21. Set the PDATA_AND_MOVER protocol - pmover.....	3-40
3.2.22. Set the Socket Buffer Size - setsockbufsize.....	3-41
3.2.23. Set the Transfer Buffer Size - setxferbufsize.....	3-42
<b>Chapter 4. User Utilities .....</b>	<b>4-1</b>
4.1. Utilities.....	4-1
4.1.1. HPSS ACL Editor - hacl .....	4-1
4.1.2. List information about HPSS - lshpss .....	4-12
<b>Acronyms .....</b>	<b>A-1</b>
<b>References .....</b>	<b>B-1</b>

---

## Chapter 1. Overview

The High Performance Storage System (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 for large storage capacities, and to use network-connected storage devices to transfer data at rates up to multiple gigabytes per second. Listed below are the user interfaces for accessing data from HPSS.

### 1.1. User Interfaces

#### 1.1.1. File Transfer Protocol (FTP)

HPSS provides an industry-standard FTP user interface. Because 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 simply means that the FTP 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, as in most other storage systems, but by the speed of the data path between the HPSS FTP daemon and the user's FTP client.

All FTP commands are supported or properly rejected if the HPSS Parallel FTP Daemon does not implement a specific feature. In addition, the ability to specify Class of Service for is provided via the **quote site** or **site** commands. Additional site command options are provided for **chgrp**, **chgid**, **chmod**, **chown**, **chuid**, **stage**, **wait**, and **symlink**. The HPSS FTP Daemon supports access from any RFC-0959 conformant FTP Client. In addition, the **quote allo64** command is supported.

Passive connections are not supported. Also, to avoid confusion, the user may want to explicitly specify the data transfer type of **ascii** or **binary**.

Refer to the HPSS System Administration Guide for information on configuring PFTP.

#### 1.1.2. Parallel FTP (PFTP)

The PFTP supports normal FTP plus extensions. It is built to optimize FTP performance for storing and retrieving files from HPSS by allowing the data to be transferred in parallel to the client. The interface pro-

---

vided to the user has syntax similar to FTP but with some extensions to allow the user to transfer data to and from HPSS across parallel communication interfaces. PFTP supports transfers either via TCP/IP. The FTP client communicates directly with HPSS Movers to transfer data.

The following constraints are imposed by PFTP.

- Pipes are not supported.
- Passive connections are not supported.
- ASCII transfers are not supported over the parallel interface because ASCII transfers insert characters. This makes it impossible to send the data in parallel. Since extra characters are inserted in the stream, there is no way to resolve data placement. Warning: Some FTP implementations default to ascii. If this is the case, it will be necessary to specify binary by entering the bin command.
- PFTP client access is supported only from nodes which support the HPSS PFTP client software.

Refer to the HPSS System Administration Guide for information on configuring PFTP.

### **1.1.3. Network File System Version 3 (NFS)**

The purpose of the NFS V3 server interface is to provide transparent access to HPSS name space object and bitfile data for client systems. Following a mount on the HPSS file system name, the user may access HPSS files using standard function calls and command interfaces.

The code written to implement the NFS V3 Server interface is written to the *Network File System Specification, RFC-1813*, DDN Network Information Center, SRI International, Menlo Park, Ca.

The NFS V3 Server interface and data structures are defined by RFC-1813.

Since there are no extensions or modifications to the NFS user interface, no additional interface information is provided in the remainder of this document.

The following constraints are imposed by the HPSS NFS V3 server.

- All files created using NFS are stored in a single Class of Service. The Class of Service used by NFS is defined by the HPSS administrator.
- NFS transfers are slower than the other HPSS interfaces, and are therefore not recommended for

---

large file accesses.

Refer to the HPSS Administration Guide for information on configuring NFS V3.

### **1.1.4. Distributed File System (DFS)**

HPSS has an option that enables it to interface with Transarc's DFS to provide distributed file system services. If a site opts to use DFS, the standard DFS interfaces can be used to administer filesets<sup>1</sup> and access the objects within the DFS filesets. Additional HPSS administrative set-up is required for HPSS to "know" about DFS filesets.

In addition to providing a distributed file system, high-speed I/O performance is essential to many sites. Since DFS could not be modified, without significant cost, to provide a high-speed I/O interface, two HPSS data management configuration options, at the fileset level, are available. Although these options are defined per fileset, it must be noted that because of limitations in the implementation, **all** filesets on a particular aggregate<sup>2</sup> must be configured with the same option.

#### **Archived Filesets:**

If the objects in a fileset will only be accessed through DFS interfaces, the fileset should be managed with the archived fileset option. HPSS is used strictly as an archive facility for DFS, with this option. Any DFS file data that has not been accessed recently will be migrated to HPSS, purged from DFS, and restored to DFS upon future access. At any given time, the data in a DFS file may be out of date with the data in the HPSS file. Clients may not use the HPSS interface to access objects in archived filesets. This option is well suited to applications that run directly on DFS, require more disk space than is available to DFS, and do not need a high-speed I/O interface.

Archived filesets incur some overhead when files are created and deleted. The overhead occurs because some processing is required to set state indicating a new file exists that must eventually be migrated to HPSS. When a file is unlinked some processing is also required to determine if the file must be marked for subsequent removal from HPSS. Removing the file from HPSS will take place at a later time. Delaying the file removal from HPSS allows for the file to be restored to DFS, if so desired.

#### **Mirrored Filesets:**

- 
1. Filesets are an administrative entity defined by DFS. They are simply a collection of files, or a directory subtree.
  2. Aggregate is a term defined by DFS, and refers to an allocation of disk space.

---

If the high-speed I/O interface to file data is required, the fileset must be managed with the mirrored fileset option. Typically, such applications would write large amounts of data to a file through the HPSS interface, and later, read smaller pieces of this data through DFS. With this option, consistency between the HPSS and DFS name and data spaces is maintained. Name or data space updates made through the DFS interface will be visible through the HPSS interface, and vice versa. A user may access data through DFS, at typical DFS rates, and when high performance I/O rates are important, use the HPSS high-speed interface.

Mirrored filesets incur additional overhead for any name space activity that alters the name space, such as, creates, unlinks, renames, and owner or permissions changes. Name space activity that does not alter the name space, such as, “ls” and “cd”, perform at typical DFS rates.

#### **Usage Considerations:**

The overhead associated with DFS client I/O to files is insignificant for both mirrored and archived filesets, unless the data must be cached from HPSS to DFS. Moving data between DFS and HPSS is highly dependent on disk speed, network speed, and system load, but, in general, our data transfers between DFS and HPSS were as fast as permitted by the hardware.

Both archived and mirrored filesets support whole or partial file caching of data from HPSS to DFS. The caching option applies to **all** filesets on an aggregate.

It is critical to understand the differences in the HPSS functionality that is provided by archived and mirrored filesets. For example, with mirrored filesets DFS and HPSS name spaces are identical, and any name space changes made through either interface are immediately visible through both the DFS and HPSS interfaces. With archived filesets, there is no correspondence between the DFS and HPSS name space. It is important to configure the fileset type to suit the behavior expected of the files in the fileset.

Since there are no extensions or modifications to the DFS user interface, no additional interface information. Refer to the HPSS Administration Guide for information on configuring DFS.

### **1.1.5. User Utilities**

The purpose of the HPSS user utilities is to provide the end user with information such as Access Control List (ACL) definitions and Class of Service definitions. In addition, the ability for a user to change his ACL definitions is provided.

The user utilities consist of these commands:

- 
- **hacl** edit HPSS Access Control Lists (note that this utility replaces the lsacl and chacl utilities)
  - **lshpss** list information for HPSS (Class of Service list, hierarchy list, storage class list, physical volumes, devices and drives, servers, Movers, and metadata)

## **1.2. Storage Concepts**

This section defines key HPSS storage concepts which have a significant impact on the usability of HPSS. Configuration of the HPSS storage objects and policies is the responsibility of your HPSS administrator.

### **1.2.1. Class of Service**

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 and the class 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 HPSS.

For the FTP and PFTP interfaces, the COS ID may be explicitly specified by using the **site setcos** command. If not specified, a default COS is used. For NFS, all files are created in the same COS. This COS is defined by your system administrator. For DFS, COS may be assigned by explicitly assigning a COS in the fileset definition. Otherwise, the size of the file is used as hints for COS selection. Contact your HPSS administrator to determine the COSs which have been defined. The **lshpss -cos** command may also be used to list the defined COSs. Refer to Chapter 5 for information on the **lshpss** command.

Also, PFTP provides a feature to automatically store the local file size in the minimum and maximum file size fields of the COS. This feature is also provided for FTP clients which support the ALLO command. This allows the COS selection to be made according to file size. The HPSS administrator should ensure that COS definitions contain proper minimum and maximum file sizes in order for PFTP (FTP clients which support ALLO) to optimize storage utilization when transferring files to HPSS. Note: If the COS ID is explicitly set by using the **site setcos** command, that COS will be used regardless of file size.

A COS is implemented by a Storage Hierarchy of one to many Storage Classes. Storage Hierarchies and Storage Classes are not directly visible to the user, but are described below since they map to COS. The relationship between storage class, storage hierarchy, and COS is shown in Figure 1-1.

---

### **1.2.2. Storage Class**

An HPSS Storage Class is used to group storage media together to provide storage with specific characteristics for HPSS data. The attributes associated with a storage class are both physical and logical. Physical media in HPSS are called physical volumes. Physical characteristics associated with physical volumes are the media type, block size, the estimated amount of space on volumes in this class, and how often to write tape marks on the volume (for tape only). Physical media are organized into logical virtual volumes. This allows striping of physical volumes. Some of the logical attributes associated with the storage class are virtual volume block size, stripe width, data transfer rate, latency associated with devices supporting the physical media in this class, and storage segment size (disk only). In addition, the storage class has attributes that associate it with a particular migration policy and purge policy to help in managing the total space in the storage class.

### **1.2.3. Storage Hierarchy**

An HPSS storage hierarchy consists of multiple levels of storage with each level representing a different storage media (i.e., a storage class). Files are moved up and down the storage hierarchy via stage and migrate operations, respectively, based upon storage policy, usage patterns, storage availability, and user request. 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 on the migration policy and staging operations. Multiple copies of a file may also be specified in the migration policy. If data is duplicated for a file at multiple levels in the hierarchy, the more recent data is at the higher level (lowest level number) in the hierarchy. Each hierarchy level is associated with a single storage class.

### **1.2.4. File Family.**

A file family is an attribute of an HPSS file that is used to group a set of files on a common set of tape virtual volumes. Release 4.1 supports grouping of files only on tape volumes. In addition, families can only be specified in Release 4.1 by associating a family with a fileset, and creating the file in the fileset. 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 family is associated with the file, the file is migrated to the next available tape not associated with a family (actually to a tape associated with family zero). If no tape virtual volume is associated with the family, a blank tape is reassigned from family zero to the file's family. The family affiliation is preserved when tapes are repacked. Configuring file families is a System Administrator function.

---

### **1.3. Interface Usage Considerations**

Guidance on when to use a particular HPSS interface is provided below. In general, PFTP provides the best data transfer performance. NFS is the slowest interface, and should not be the interface of choice for large HPSS data transfers.

Conditions in which the user might elect to use FTP are:

- Utilizes standard FTP interface

Users and applications familiar with FTP can access HPSS with the standard command set.

- Supports files greater than 2 gigabytes

FTP supports file sizes up to  $2^{64}$ .

- Supports any FTP client platforms

FTP commands may be issued from any vendor nodes with an FTP interface. No specialized code is required.

Conditions in which a user might elect to use PFTP are:

- Provides faster file transfers.

PFTP is a better performer than FTP since it provides the capability to stripe data across multiple client data ports.

- Supports files greater than 2 gigabytes

PFTP supports file sizes up to  $2^{64}$ .

- Supports partial file transfer.

PFTP provides options on the pget and pput commands to perform partial file transfers. This would be beneficial to users who want to extract pieces of large files.

Conditions in which a user might elect to use NFS are:

- Provides standard system access

Files may be accessed and managed through standard system mechanisms without calling a special library or program to translate commands

- 
- Eliminates multiple file instances

The need to maintain multiple instances of a file can be eliminated since files remain on the NFS server.

- Accesses limited to smaller files
- Supports any NFS client platforms

NFS access is supported from any client nodes with an NFS V2 or V3 interface. No specialized code is required.

Conditions in which a user might elect to use DFS are:

- Provides standard system interface

Files may be accessed and managed through standard system mechanisms without calling a special library or program to translate commands

- Supports any DFS client platforms

DFS access is supported from any vendor nodes with aDFS client interface.

- Supports files larger than 2 gigabytes.

- Small file performance.

Provide DFS transfer rates for smaller files (unless migrated to HPSS)

- Other

DFS provides security and global name space features

Extend DFS storage through a back-end to mass storage.

Provide mirrored option which supports data accesses from either the DFS interface or any of the HPSS interfaces. This would be used for large files with higher performance demands.

## **1.4. User IDs**

After the HPSS system is configured, the necessary accounts must be created for HPSS users. Contact your HPSS administrator to add an account.

---

For FTP / PFTP access, an FTP account must be created. The administrator can use the **hpssuser -add user -ftp** command to add a new FTP user.

For NFS access, a DCE user account must be created. The administrator can use the **hpssuser -add user -dce** command to add a new DCE account.

For DFS access, the user must have a DCE account and be logged in to DCE.

Users calling the utilities described in this document must be logged into DCE. As noted above, the administrator can use the **hpssuser -add user -dce** command to add a new DCE account.

## **1.5. DCE User Accounts**

As mentioned in the previous section, the user utilities and DFS require the user be logged into DCE.

The following command is used to issue a DCE login:

```
dce_login [principal_name] [password]
```

When this command is entered, the principal's identity is validated, and the network credentials are obtained. If *principal name* or *password* are not supplied, **dce\_login** will prompt for them.

When the principal's DCE login context is no longer required, the following command may be used to destroy the login context and associated credentials:

```
kdestroy
```

Other DCE commands which might be of interest to the user are:

```
klist list the primary principal and tickets held in the DCE credentials cache
```

```
kinit Refresh a DCE credentials cache
```



## Chapter 2. File Transfer Protocol (FTP)

This chapter specifies the HPSS FTP interface. For information on configuring the HPSS FTP daemon, reference the *HPSS Administration Guide*. FTP is supported from any FTP client platform.

HPSS supports the FTP command set for transferring files to and from HPSS. To use FTP, the user enters the following:

```
ftp node_name [port_number]
```

where,

*node\_name* is the node name of the node where the HPSS FTP Daemon process resides

*port\_number* is the port number for HPSS, as set up in `/etc/services`

At this point, any standard FTP command may be entered. Note: If the message "Load thread state failed" is received, contact your HPSS administrator. This message generally implies that either HPSS is not correctly configured, or some HPSS components may not be executing:

### 2.1. Site Commands

HPSS also supports the site commands listed below, e.g. **site setcos 300** or **quote site setcos 300**.

Note: On some platforms, it may be necessary to specify **quote site** instead of **site**.

- **setcos**
- **chgid**
- **chgrp**
- **chmod**
- **chown** (valid only for "root" account)
- **chuid**
- **stage**
- **wait**
- **symlink**

### **2.1.1. Specifying a File's Class of Service - setcos**

setcos is used to specify a class of service and has the following format:

```
quote site setcos cos_id
```

where,

*cos\_id* is the Class of Service identifier (used when creating a new HPSS file during a put operation.)

Class of Service is used as a means for specifying the amount of parallelism or stripe width for a file. See your HPSS system administrator for the Class of Service identifiers defined for your site. If a Class of Service is not specified, a default is used.

In the example below, the following commands might be entered to put a large file to HPSS with a Class of Service identifier of 4. In this example, 4 might designate 4-way striping to 3490 tape.

```
quote site setcos 4
```

### **2.1.2. Changing a File's Group by ID - chgid**

chgid is used to change the group ID of a file and has the following format:

```
quote site chgid gid file
```

where,

*gid* is the new group ID of the file

*file* is the name of the file.

The user must belong to the specified group and be the owner of the file, or be the root user.

Example: The following may be entered to change the group ID of myfile to group ID 210.

```
quote site chgid 210 myfile
```

### **2.1.3. Changing a File's Group by Name - chgrp**

chgrp is used to change the group name of a file and has the following format:

```
quote site chgrp group file
```

where,

*group* is the new group name of the file, and *file* is the name of the file.

The user must belong to the specified group and be the owner of the file, or be the root user.

Example: The following may be entered to change the group of myfile to group mygroup.

```
quote site chgrp mygroup myfile
```

### **2.1.4. Changing a File's Permissions - chmod**

chmod is used to change the mode of a file and has the following format:

```
quote site chmod mode file
```

where,

*mode* is the new octal mode number of the file

*file* is the name of the file.

Mode is constructed from the OR of the following modes:

0400	read by owner
0200	write by owner
0100	execute (search in a directory) by owner
0040	read by group
0020	write by group
0010	execute (search in a directory) by group
0004	read by others
0002	write by others
0001	execute (search in a directory) by others

Note: The following mode values are not supported:

4000	set user ID on execution
2000	set group ID on execution
1000	sticky bit

Only the owner of the file or root user can change its mode.

Example: The following may be entered to change the mode of myfile to read, write by owner and group.

```
quote site chmod 0660 myfile
```

### **2.1.5. Changing a File's Owner by Name - chown**

chown is used to change the owner of a file and has the following format:

```
quote site chown owner file
```

where,

*owner* is the new owner of the file

*file* is the name of the file.

Only the root user can change the owner of a file.

Example: The following may be entered to change the owner of /home/smith/myfile to jones.

```
quote site chown jones /home/smith/myfile
```

### **2.1.6. Changing a File's Owner by ID - chuid**

chuid is used to change the uid of a file and has the following format:

```
quote site chuid uid file
```

where,

*uid* is the new uid of the owner of the file

*file* is the name of the file.

Only the root user can change the uid of a file.

Example: The following may be entered to change the uid of /home/smith/myfile to 201.

```
quote site chuid 201 /home/smith/myfile
```

### 2.1.7. Staging a File - stage

stage is used to initiate a stage of a migrated file (e.g. from tape to disk). The user can initiate the stage and then return at a later time to initiate the file transfer using the FTP **get** or PFTP **pget** commands:

```
quote site stage file
```

where,

*file* is the name of the file.

Example: The following may be entered to stage file /home/smith/myfile.

```
quote site stage /home/smith/myfile
```

### 2.1.8. Setting the Desired Wait Options (for Migrated Files) - wait

wait is used to notify the HPSS PFTP Daemon :

```
quote site wait option
```

where,

*option* is one of the following values:

-1 or inf(inite) - wait forever for the file to be staged. Do not return from the **get** / **pget** command to complete until the file has been transferred or a transfer error has occurred.

0 - do not wait for the file to be staged. If the file has been migrated, return the appropriate message and initiate the stage. The user will return later to reissue the **get** / **pget** command.

n (where n is an integer) - wait the specified period (in seconds) for the file requested by a **get** / **pget** command to complete. Either transfer the file if the file is staged within the specified period or return a reply to notify the user to try again later.

Example: The following may be entered to wait for files to be staged.

```
quote site wait -1
```

The following table describes the behaviour the customer should expect from FTP when issuing the stage/wait commands. Note: ONLY Classes of service utilizing the "Stage on Background" will exhibit predictable results.

Stage/Wait Behaviour

Wait Time	File Condition	Command	Behaviour /Message
No Wait	Archived	quote site stage xyz	“File xyz is being retrieved from archive.”
No Wait	Not Archived	quote site stage xyz	“File xyz is currently ready for other processing.”
Wait ###	Archived	quote site stage xyz	Wait for period then receive message: “File xyz is currently ready for other processing.” or “File xyz is currently ready for other processing.” if the file is staged in the time frame allowed
Wait ###	Not Archived	quote site stage xyz	“File xyz is currently ready for other processing.”
No Wait	Archived	get xyz	“File xyz is being retrieved from archive.”
No Wait	Not Archived	get xyz	Transfers Data as expected.
Wait ###	Archived	get xyz	Wait for period then receive message: “File xyz is being retrieved from archive.” or transfers data as expected if file is staged in the time allowed.
Wait ###	Not Archived	get xyz	Transfers file as expected.

### 2.1.9. Creating a Symbolic Link - symlink

symlink is used to create a symbolic link.

**quote site symlink *path/file link***

where,

*path/file* refers to the destination

*link* refers to the local filename.

Example: The following may be entered to create a link names sys\_passwd in the local directory pointing to /etc/passwd.

**quote site symlink /etc/passwd sys\_passwd**

A dir command will show sys\_passwd -> /etc/passwd.

### 2.1.10. Allocating space for files - quote allo64

The quote allo64 command is used to specify the size of a file for space allocation.

**quote allo64 *size***

where,

*size* is a string representing the size of the file. The size may be a decimal number less than  $2^{64}$  or may be in the form 1MB (1048576). No spaces are allowed between the decimal number and the magnitude representation string. Accepted magnitude representation strings are:

KB (kilobyte = 1024),

MB (megabyte = 1048576),

GB (gigabyte = 1073741824),

TB (terabyte = 1099511627776),

PB (petabyte = 1125899906842624).

The magnitude representation string is case independent. The decimal component may contain up to two decimal points of precision. NOTE: 1005.03 will truncate to 1005 if no magnitude representation string is specified. Similar truncations will occur for excess precision specifications.

This command provides a 64-bit extension to the standard **quote allo size** command. NOTE: the **quote allo size** command only accepts decimal values for size. Both these commands are helpful for providing hints for non-parallel "put" commands.

Example: The following may be entered to specify the file size of 8 gigabytes.

```
quote allo64 8GB
```

## 2.2. List Directory Extensions

FTP supports the **ls** command to list the contents of a directory. Standard options supported are: **ls**, **ls -l**, **ls -a**, and **ls -F**. In addition to the standard **ls** options generally provided, HPSS also provides a **-lh** option. If **-lh** is specified, then a long directory listing is generated. However, in place of the owner field (field #3) and group field (field #4) listed for the **-l** option, the Class of Service identifier and Account Code are listed.

Example: **ls -lh**

```
-rw-rw---- 1 1    198 157286400 May 13 1996 TEST
-rw-r--r-- 1 1    160   32768 May 16 1996 prod1
-rw-r--r-- 1 1    160   32768 May 16 1996 prod10
-rw-r--r-- 1 1    160   32768 May 16 1996 prod11
-rw-r--r-- 1 1    160   32768 May 16 1996 prod12
-rw-r--r-- 1 1    160   32768 May 16 1996 prod13
-rw-r--r-- 1 1    160   32768 May 16 1996 prod14
-rw-r--r-- 1 1    160   32768 May 16 1996 prod15
-rw-r--r-- 1 1    160   32768 May 16 1996 prod151
-rw-r--r-- 1 1    160   32768 May 16 1996 prod152
```

-rw-r--r--	1 1	160	32768	May 16 1996	prod153
-rw-r--r--	1 1	160	32768	May 16 1996	prod154
-rw-r--r--	1 1	160	32768	May 16 1996	prod155
-rw-r--r--	1 1	160	32768	May 16 1996	prod156
-rw-r--r--	1 1	160	32768	May 16 1996	prod157
-rw-r--r--	1 1	160	32768	May 16 1996	prod158
-rw-r--r--	1 1	160	32768	May 16 1996	prod159

## Chapter 3. Parallel File Transfer Protocol (PFTP)

This chapter specifies the HPSS PFTP interface. For information on configuring the HPSS PFTP daemon, reference the *HPSS Administration Guide*. In order to use PFTP, the PFTP client code must be compiled and supported on the client platform.

PFTP supports the FTP command set plus some additional commands (refer to the next subsection). To use PFTP, the user enters one of the following commands:

```
pftp_client [-bStringSize] [-c] [-d] [-e] [-g] [-h] [-i] [-m] [-n] [-p] [-t] [-v] [-w###] [-BStringSize] [-C###] [-Rstring] [-SsizeString] [Host [Port]]
```

```
krb5_gss_pftp_client [-bStringSize] [-c] [-d] [-e] [-g] [-h] [-i] [-m] [-n] [-p] [-t] [-v] [-w###] [-BStringSize] [-C###] [-Rstring] [-SsizeString] [Host [Port]]
```

where,

- b Sets the PDATA protocol Blocksize. StringSize is the size specification in the format: Digit(s)Magnitude; e.g. 1MB.
- c Sets “Child” mode. This provides the ability to “emulate” a tty and interactive mode when executing the client in a “batch” mode.
- d The standard FTP debug specification.
- e Sets “Echo” mode. When running the client in batch mode, this causes the client to echo each command into the output file providing a helpful record of commands interleaved with the return messages.
- g Disables the expansion of metacharacters in file names. Interpreting metacharacters can be referred to as expanding (sometimes called globbing) a file name. See the glob subcommand.
- h Specific to use the original HPSS protocol, PDATA\_AND\_MOVER, regardless of the default specified in the HPSS.conf file
- i Turns off interactive prompting during multiple file transfers. See the prompt, mget, mput, and mdelete subcommands for descriptions of prompting during multiple file transfers.
- k Kerberos ONLY option to specify an alternate Kerberos Realm for the PFTP Daemon.
- m This argument will enable multinode processing. By default, multinode processing is disabled. Multinode will be ignored if NO multinode specification for this client / daemon pair is specified in the HPSS.conf file.
- n Prevents an automatic login on the initial connection. Otherwise, the ftp

	command searches for a \$HOME/.netrc entry that describes the login and initialization process for the remote host. See the user subcommand.
-p	Specifies to use the HPSS protocol (PDATA_ONLY) for parallel transfers regardless of the default in the HPSS.conf file.
-t	The standard FTP trace specification.
-v	Displays all the responses from the remote server and provides data transfer statistics. This display mode is the default when the output of the ftp command is to a terminal, such as the console or a display.
-w	This argument will set the pwidth. The pwidth value must be specified immediately following this argument.
-B	Sets the Parallel Blocksize for parallel transfers. StringSize is the size specification in the format: Digit(s)Magnitude; e.g. 1MB.
-C	Sets the default Class of Service (COS) for the session. The argument is a valid string representation of a decimal COS. COS names are NOT accepted.
-R	Used to specify the valid ports for parallel transfers. This is useful in instances where network filters are invoked which provide port ranges for TCP traffic. The syntax is "start_range-end_range". These values are translated into the environment variable "HPSS_PFTPC_PORT_RANGE=ncacn_ip_tcp[start_range-end_range]" environment variable which is parsed by the client as necessary.
-S	Sets the maximum open / close socket size for HPSS parallel transfers. An artificial maximum of 250GB (subject to change) is compiled in. Results may be smaller than the specified value based on a number of external HPSS constraints. The default is 1.5GB. StringSize is the size specification in the format: Digit(s)Magnitude; e.g., 200GB.
Host	The node where the HPSS FTP Daemon process resides.
Port	PortThe port number for the HPSS ftpd, as set up in /etc/services.

The local administrator may opt to define a **pftp** program link that points to **pftp\_client**.

An additional variant of the **pftp\_client**, **krb5\_gss\_pftp\_client** may be built by the customer site. Contact your site representative for details. These clients utilize the MIT Kerberos GSS facilities for authentication and reply processing. The GSS-based clients are used to provide credential authentication facilities (password-less authentication) between the client and the HPSS GSS Parallel FTP Daemon using either Kerberos or DCE credentials for authentication. Since the Generic Security Service (GSS) versions of the Parallel FTP Client only relate to the authentication process, these clients should behave identical to the non-GSS versions after authentication. MIT Kerberos is available from MIT and will NOT be supplied by the HPSS project. NOTE: The HPSS (GSS) Parallel FTP Client and Daemon are incompatible with the Kerberos-based FTP features provided by IBM with AIX 4.x. The HPSS (GSS) Parallel FTP Client and Daemon are compatible with the MIT FTP processes.

The GSS version of the Parallel FTP Client requires MIT Kerberos and/or compatible Client software (headers / libraries). Neither IBM nor the HPSS development team are obligated to continue the GSS PFTP in the future.

As a courtesy to HPSS customers, the Parallel FTP Client code is available for compilation at customer sites upon request. The Parallel FTP Client code will be provided as a `clients_port.tar.Z` file (tarred and compressed) containing all components required to build the applications. Hardware/Software dependencies are the individual HPSS customers responsibility. This explicitly denies any support requirement on IBM or the HPSS Development/Support personnel for any modifications made by the customer. No DCE software is required to build the HPSS Parallel FTP Client.

The HPSS PFTP Client has been successfully compiled on: Cray UNICOS, Hewlett-Packard HPUX (32 / 64 Bit), Silicon Graphics IRIX (32 / 64 Bit), Sun Solaris (32 /64 Bit), Intel Paragon OSF (discontinued), Intel Teraflop OSF, Linux Intel (32 / 64 Bit), Compaq Alpha, and IBM AIX 4.x (32 / 64 Bit). Ports to other hardware/software components are the responsibility of the remote site. These sites will be asked to share their ports with the HPSS development team (and other HPSS facilities); however, neither IBM nor the HPSS Development Team accepts any obligation to incorporate any hardware/software ports into the distribution source. No site specific features (local mods) added to the Parallel FTP client by customer sites will be incorporated into the PFTP client without the modification of the HPSS license.

The GSS versions of the Parallel FTP Client, `krb5_gss_pftp_client`, require MIT Kerberos. Neither IBM nor the HPSS development team declare the Kerberos code suitable for any specific purpose nor are they obligated to repair or support customers using this code.

The GSS HPSS Parallel FTP Daemon, `hpss_pftp_amgr` and `auth_krb5gss` executables, are available for IBM AIX 4.x and Solaris 8.

Note: If the message "Load thread state failed" is received, contact your HPSS administrator. This message generally implies that either HPSS is not correctly configured, or some HPSS components may not be executing.

### **3.1. Parallel FTP Client Transfers**

Parallel transfers involve the creation of child processes to transfer the data between the source and the destination. This process may be either local spawned PFTP client children, remotely initiated PFTP "children," or the combination of both. When the `pwidth` value is set and a valid multinode configuration file does not exist or multinode has not been activated, the PFTP client will provide parallel data paths to the Movers by spawning multiple processes on the client node using one or more network interfaces (NICs).

The multinode option supports spawning the client processes across multiple machines / nodes and/or multiple interfaces on the remote machines / nodes. This multinode option may be beneficial on processors which support shared file systems, such as GPFS on the IBM SP. Note: if multinode is used in a non-shared file system, the multinode file transfer to the client will be spread across multiple, separate files, which is not the desired behaviour. The client nodes which participate in a multinode transfer are selected from the `HPSS.conf` configuration file which contains entries with control, and optionally, data interface names or addresses. The number of nodes selected from the configuration file is based on the `pwidth` value. The starting node is selected using an offset of which is maintained by the PFTP client.

### 3.1.1. Parallel FTP Client Configuration

The PFTP client requires configuration in the HPSS.conf file to provide optimal performance characteristics.

#### 3.1.1.1. HPSS.conf File

This configuration file is used to specify performance optimization parameters for the PFTP components, the HPSS Movers, and potentially Site specific applications. Ref: HPSS.conf(7).

#### 3.1.1.2. PFTP Specific HPSS.conf entries

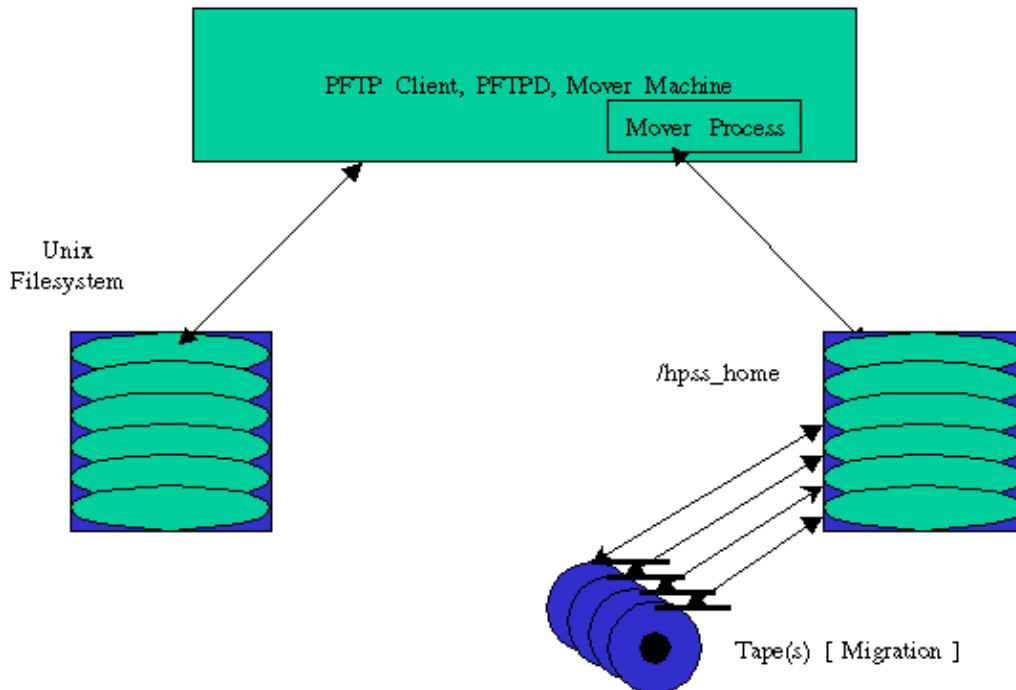
Reference the HPSS.conf(7) man page.

#### 3.1.1.3. Local File Functions

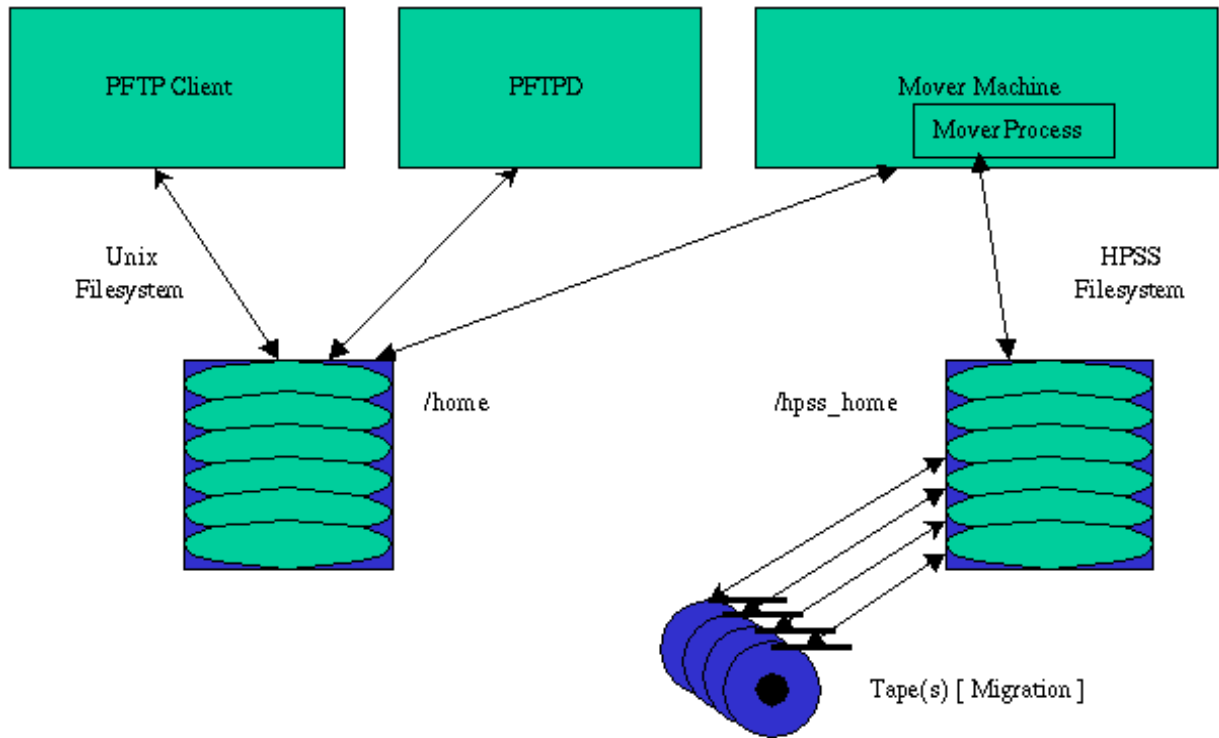
The Local File Functions represent performance enhancements using the HPSS parallel protocols where both the HPSS file and the Unix source/destination file are "globally available" to the mover(s) and the PFTP client processes (e.g., GPFS filesystems.) The local file path MUST be specified in the file: /var/hpss/etc/hpss\_mvr\_localfilepath.conf (FUTURE?: the "Local File Path" specification will be specified in the HPSS.conf file.) The specified path specified MUST exist for each PFTP Client/Mover machine.

#### Configuration Criteria:

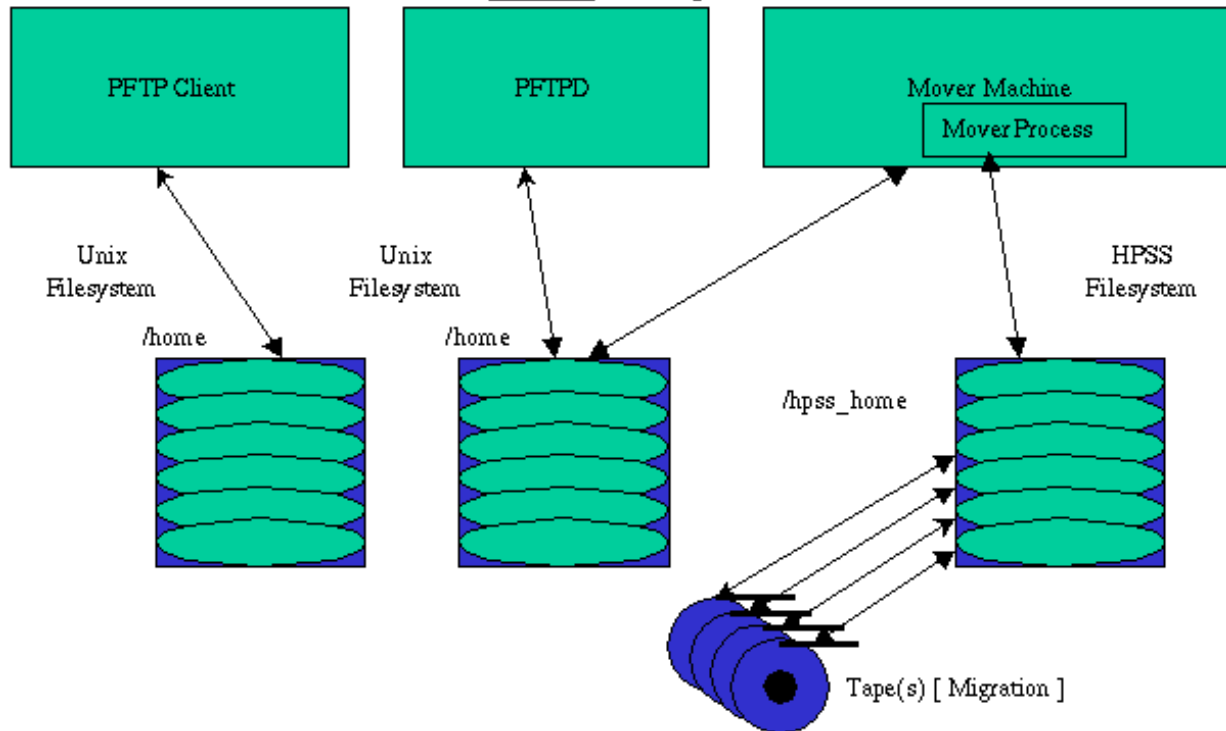
Local File Transfer Optimal Configuration for Parallel PFTP



Local File Transfer Probable Configuration for Parallel FTP



Local File Transfer Invalid Configuration for Parallel FTP



### 3.2. Additional HPSS Commands

All FTP extensions described in Chapter 2 are supported by PFTP. In addition, the following commands (abbreviations) are supported by PFTP.

·	pappend	(papp)
·	pput, mpput	(ppu, mpp)
·	pget, mpget	(pge, mpg)
·	lfappend	(lfa)
·	lfput, mlfp	(lfp, mlfp)
·	lfget, mlfg	(lfg, mlfg)
·	psocket	(psock)
·	setpwidth	(setpw)
·	setpblocksize	(setpb)
·	multimode	(multi)
·	autoparallel	(autop)
·	getprot	(getp)
·	gettuningparms	(gettun)
·	pdata	(pdat)
·	penable	(pen)
·	pmover	(pmov)
·	setsockbufsize	(sets)
·	setxferbufsize	(setx)

NOTE:

**Pipes are NOT supported for the pget and pput commands** even if the HPSS.conf specifies a valid pipe directory. Ref: HPSS.conf(7). This is the result of external timing problems which make this operation unreliable. To use pipes it is mandatory that you specify the **autop** command to **disable** automatic substitution of parallel commands and then explicitly use the “get” or “put” commands!

**3.2.1. General Login messages (Examples)**

```
Connected to water.clearlake.ibm.com.
220-#
220-#  HPSS Parallel FTP Daemon on water
220-#
220-
220 water FTP server (HPSS 4.3 PFTPD V1.1.4 Tue May 29 14:55:58 CDT 2001) ready.
Name (water:whrahe):
331 Password required for /.../water_cell.clearlake.ibm.com/whrahe.
Password:
230 User /.../water_cell.clearlake.ibm.com/whrahe logged in.

Remote system type is UNIX.
Using binary mode to transfer files.
**** NOTE: Server supports Parallel Features ****
****   Auto-Parallel Substitution Enabled. ****
**** NOTE: Protocol set to PDATA_AND_MOVER ****
Multinode is Disabled.

ftp>
```

### 3.2.2. Parallel append - pappend

#### Synopsis

```
pappend local_file [remote_file]
```

#### Description

The **pappend** command transfers a file from the local machine to HPSS. The transfer starts at the end of the remote file and continues until the entire file is moved or until an error occurs.

#### Parameters

<i>local_file</i>	Identification of the file to transfer on the local machine.
<b>remote_file</b>	Optional file name to the remote file. If not supplied then the remote (HPSS) file name defaults to be the same as the local file name.

#### Return strings

Output shows the amount of data transferred and any error conditions.

#### Error conditions

Connection Failures: data transfer connection malfunction.

Network Failures: data transfer malfunction.

Allocation Failures: no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error codes are:

-5 - an I/O error occurred.

-28 - no space remaining in the associated storage class.

#### See also

RFC-0959.

#### Notes

none.

#### Examples

1. Append local file testfile to the same file name in the user's HPSS home directory.  

```
ftp> pappend testfile
```
2. Append local file testfile to HPSS file prod1 in the current working directory.

ftp> **pappend testfile prod1**

### 3.2.3. Parallel file store - pput

#### Synopsis

```
pput [-l local_offset] [-r remote_offset] [-s size] local_file  
[remote_file]
```

#### Description

The **pput** command transfers a file from the local machine to HPSS. If offsets and size of transfer are not specified, the transfer starts at the beginning of the local file and continues until the entire file is moved or until an error occurs. However, flexibility is provided to perform partial file transfers by specifying local file offset, remote file offset, and size of transfer. The *local\_offset*, *remote\_offset*, and *size* may be specified using a decimal and magnitude representation string. See Section 2.1.10 for use of this notation.

The normal **pput** command functions just like the standard ftp **put** command and transfers an entire file.

#### Parameters

<i>-l local_offset</i>	Optional byte offset into the local file where the transfer is to begin.
<i>-r remote_offset</i>	Optional byte offset into the remote file where the data is to be placed.
<i>-s size</i>	Optional byte size of the amount of data to transfer.
<i>local_file</i>	Identification of the file to transfer on the local machine.
<i>remote_file</i>	Optional file name to the remote (HPSS) file. If not supplied then the remote file name defaults to be the same as the local file name.

#### Return strings

Output shows amount of data transferred and any error conditions.

#### Error conditions

Connection Failures:	data transfer connection malfunction.
Network Failures:	data transfer malfunction.
Allocation Failures:	no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error codes are:

- 5 - an I/O error occurred.
- 28 - no space remaining in the associated storage class.

#### See also

RFC-0959.

### Notes

none.

### Examples

1. Transfer local file testfile to the user's HPSS home directory.

```
ftp> pput testfile
```

2. Transfer local file testfile to HPSS file prod1 in the current working directory.

```
ftp> pput testfile prod1
```

3. Transfer 1MB from offset 1MB of local file testfile to offset 0 of HPSS file /home/bob/prod1.

```
ftp> pput -l 1048576 -r 0 -s 1048576 testfile /home/bob/prod1.
```

4. Transfer all local files which begin with "test" to the user's HPSS home directory using a pipe and tar (bundling). **Parallel Pipes are NOT supported!** Specify the "autop" command to disable automatic parallel command substitution nad use the "put" command!

```
ftp> pput " | tar cf - ./test*" my_test.tar
```

### 3.2.4. Parallel file store - mpput

#### Synopsis

```
mpput local_files
```

#### Description

The **mpput** command expands the files specified in the *local\_files* parameter at the local host and copies the indicated files to HPSS. The **mpput** command functions just like the standard ftp **mput** command.

#### Parameters

*local\_files*                      Identification of the files to transfer on the local machine.

#### Return strings

Output shows the amount of data transferred and any error conditions.

#### Error conditions

Connection Failures:    data transfer connection malfunction.

Network Failures:        data transfer malfunction.

Allocation Failures:    no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error codes are:

-5 - an I/O error occurred.

-28 - no space remaining in the associated storage class.

#### See also

RFC-0959.

#### Notes

none.

#### Examples

1.        Transfer all local files in the current directory to the user's HPSS home directory.

```
ftp> mpput *
```

2.        Transfer all local files which begin with test in directory /usr/bob to the user's HPSS home directory.

```
ftp> lcd /usr/bob
```

```
ftp> mpput test*
```

### 3.2.5. Parallel file retrieval - pget

#### Synopsis

```
pget [-r remote_offset] [-l local_offset] [-s size] remote_file[local_file]
```

#### Description

The **pget** command transfers a file to the local machine from HPSS. If offsets and size of transfer are not specified, the transfer starts at the beginning of the remote file and continues until the entire file is moved or until an error occurs. However, flexibility is provided to perform partial file transfers by specifying remote file offset, local file offset, and size of transfer. The *local\_offset*, *remote\_offset*, and *size* may be specified using a decimal and magnitude representation string. See Section 2.1.10 for use of this notation.

The standard **pget** command transfers entire files similar to the standard **ftp** get command.

#### Parameters

<b>-r</b> <i>remote_offset</i>	Optional byte offset where transfer is to begin in the remote file.
<b>-l</b> <i>local_offset</i>	Optional parameter where the data is transferred in the local file.
<b>-s</b> <i>size</i>	Optional number of bytes to transfer.
<i>remote_file</i>	Identification of the file to transfer from the remote (HPSS) host.
<i>local_file</i>	Optional file name to the local file. If not supplied then the local file name defaults to be the same as the remote file name.

#### Return strings

Output shows the amount of data transferred and any error conditions.

#### Error conditions

Connection Failures: data transfer connection malfunction.

Network Failures: data transfer malfunction.

Allocation Failures: no space on local machine for file.

Error codes may also be returned from HPSS. The most common error code is:

-5 - an I/O error occurred.

#### See also

RFC-0959.

#### Notes

none.

### Examples

1. Transfer HPSS file /home/bob/prod1 to the user's local directory.

```
ftp> pget /home/bob/prod1
```

2. Transfer HPSS file prod1 in the current working directory to local file testfile1.

```
ftp> pget prod1 testfile1
```

3. Transfer 1MB from offset 0 of HPSS file /home/bob/prod1 to offset 1048576 of local file testfile.

```
ftp> pget -r 0 -l 1048576 -s 1048576 /home/bob/testfile1 testfile
```

4. Transfer and untar a tar file into the user's current working directory using a pipe and tar (unbundling). Parallel Pipes are NOT supported! Specify the autp command to disable automatic parallel command substitution and use the "get" command!

```
ftp> pget my_test.tar " | tar xf -"
```

### 3.2.6. Parallel file retrieval - mpget

#### Synopsis

```
mpget remote_files
```

#### Description

The **mpget** command expands the *remote\_files* parameter at the remote (HPSS) host and copies the indicated HPSS files to the current directory on the local host. The **mpget** command functions just like the standard ftp **mget** command.

#### Parameters

*remote\_files*                      Identification of the files to transfer from the remote (HPSS) host.

#### Return strings

Output shows the amount of data transferred and any error conditions.

#### Error conditions

Connection Failures:    data transfer connection malfunction.

Network Failures:        data transfer malfunction.

Allocation Failures:    no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error code is:

-5 - an I/O error occurred.

#### See also

RFC-0959.

#### Notes

none.

#### Examples

1. Transfer all files in HPSS directory /home/bob to the user's local directory.

```
ftp> cd /home/bob
```

```
ftp> mpget *
```

2. Transfer all HPSS files which begin with test in directory /home/bob to the user's local directory.

```
ftp> cd /home/bob
```

ftp> mpget test\*

### 3.2.7. Local File append - lfappend

#### Synopsis

```
lfappend local_file [remote_file]
```

#### Description

The **lfappend** is a performance optimized Parallel FTP Client command used to append a "globally available" file into HPSS using the "parallel" protocols. IF the input file is not available to the mover(s) machine (NOT "globally available") the transfer will fail because the mover(s) will NOT be able to locate the desired file. The difference between lfappend and pappend is that the mover(s) involved in the transfer, will not use the network to move the data providing improved performance. The mover(s) machine MUST be correctly configured. The file: /var/hpss/etc/hpss\_mvr\_localfilepath.conf MUST exist on each "local file" aware mover machine and MUST contain entries specifying which directories are eligible for "local file" transport.

The **lfappend** command transfers a file from the local machine to HPSS. The transfer starts at the end of the remote file and continues until the entire file is moved or until an error occurs.

#### Parameters

<i>local_file</i>	Identification of the "globally available" file to transfer.
<i>remote_file</i>	Optional file name to the remote file. If not supplied then the remote (HPSS) file name defaults to be the same as the "globally available" file name.

#### Return strings

Output shows the amount of data transferred and any error conditions.

#### Error conditions

Connection Failures: data transfer connection malfunction.

Network Failures: data transfer malfunction.

Allocation Failures: no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error codes are:

-5 - an I/O error occurred.

-28 - no space remaining in the associated storage class.

#### See also

RFC-0959.

#### Notes

none

### Examples

1. Append "globally available" file testfile to the same file name in the user's HPSS home directory.

```
ftp> lfappend testfile
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

2. Append "globally available" file testfile to HPSS file prod1 in the current working directory.

```
ftp> lfappend testfile prod1
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

**3.2.8. Local File store - lfput****Synopsis:**

```
lfput [-l local_offset] [-r remote_offset] [-s size] local_file [remote_file]
```

**Description**

The **lfput** is a performance optimized Parallel FTP Client command used to transfer a "globally available" file into HPSS using the "parallel" protocols. IF the file is not available to the mover(s) machine (NOT "globally available") the transfer will fail because the mover(s) will NOT be able to locate the desired file. The difference between lfput and pput is that the mover(s) involved in the transfer, will not use the network to move the data providing improved performance. The mover(s) machine MUST be correctly configured. The file:

`/var/hpss/etc/hpss_mvr_localfilepath.conf` MUST exist on each "local file" aware mover machine and MUST contain entries specifying which directories are eligible for "local file" transport.

If offsets and size of transfer are not specified, the transfer starts at the beginning of the local file and continues until the entire file is moved or until an error occurs. However, flexibility is provided to perform partial file transfers by specifying local file offset, remote file offset, and size of transfer. The *local\_offset*, *remote\_offset*, and *size* may be specified using a decimal and magnitude representation string. See Section 2.1.10 for use of this notation. The normal **lfput** command functions just like the standard ftp **put** command and transfers an entire file.

**Parameters**

<code>-l <i>local_offset</i></code>	Optional byte offset into the "globally available" file where the transfer is to begin.
<code>-r <i>remote_offset</i></code>	Optional byte offset into the remote file where the data is to be placed.
<code>-s <i>size</i></code>	Optional byte size of the amount of data to transfer.
<i>local_file</i>	Identification of the "globally available" file to transfer. (MUST be available to mover(s) machines)
<i>remote_file</i>	Optional file name to the remote (HPSS) file. If not supplied then the remote file name defaults to be the same as the "globally available" file name.

**Return strings**

Output shows amount of data transferred and any error conditions.

**Error conditions**

Connection Failures:	data transfer connection malfunction.
Network Failures:	data transfer malfunction..
Allocation Failures:	no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error codes are:

-5 - an I/O error occurred.

-28 - no space remaining in the associated storage class.

### See also

RFC-0959.

### Notes

none.

### Examples

1. Transfer local file testfile in the current working directory of the client to the user's HPSS home directory.

```
ftp> cd ~
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp> lfput testfile
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

2. Transfer local file testfile in the current working directory of the client to HPSS file prod1 in the current working directory.

```
ftp> lfput testfile prod1
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

3. Transfer 1MB from offset 1MB of local file testfile in the current working directory of the client to offset 0 of HPSS file /home/bob/prod1 with a new name testfile2..

```
ftp> lfput -l 1048576 -r 0 -s 1048576 testfile /home/bob/prod1/testfile2
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

### 3.2.9. Local file retrieval - **lfget**

#### Synopsis

```
lfget [-r remote_offset] [-l local_offset] [-s size] remote_file [local_file]
```

#### Description

The **lfget** is a performance optimized Parallel FTP Client command used to transfer an HPSS file into a "globally available" file using the "parallel" protocols. IF the current working directory or the specified directory is not available to the mover(s) machine (NOT "globally available") the transfer will fail because the mover(s) will NOT be able to locate the desired file. The difference between **lfget** and **pget** is that the mover(s) involved in the transfer, will not use the network to move the data providing improved performance. The mover(s) machine MUST be correctly configured. The file: `/var/hpss/etc/hpss_mvr_localfilepath.conf` MUST exist on each "local file" aware mover machine and MUST contain entries specifying which directories are eligible for "local file" transport.

The **lfget** command transfers a file from HPSS to a "globally available" directory on the mover machine(s). If offsets and size of transfer are not specified, the transfer starts at the beginning of the remote file and continues until the entire file is moved or until an error occurs. However, flexibility is provided to perform partial file transfers by specifying remote file offset, local file offset, and size of transfer. The *local\_offset*, *remote\_offset*, and *size* may be specified using a decimal and magnitude representation string. See Section 2.1.10 for use of this notation.

The standard **lfget** command transfers entire files similar to the standard **ftp** get command.

#### Parameters

<b>-r</b> <i>remote_offset</i>	Optional byte offset where transfer is to begin in the remote file.
<b>-l</b> <i>local_offset</i>	Optional parameter where the data is transferred in the local file.
<b>-s</b> <i>size</i>	Optional number of bytes to transfer.
<i>remote_file</i>	Identification of the file to transfer from the remote (HPSS) host.
<i>local_file</i>	Optional file name to the local file. If not supplied then the local file name defaults to be the same as the remote file name.

#### Return strings

Output shows the amount of data transferred and any error conditions.

#### Error conditions

Connection Failures: data transfer connection malfunction.

Network Failures: data transfer malfunction.

Allocation Failures: no space on local machine for file.

Error codes may also be returned from HPSS. The most common error code is:

-5 - an I/O error occurred.

**See also**

RFC-0959.

**Notes**

none.

**Examples**

1. Transfer HPSS file /home/bob/prod1 to the user's local directory.

```
ftp> lfget /home/bob/prod1
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

2. Transfer the HPSS file prod1 in the current working directory to local file testfile1.

```
ftp> lfget prod1 testfile1
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

3. Transfer the HPSS file testfile into the "globally available" directory /home/bob renaming the file to testfile.

```
ftp> lfget prod1 /home/bob/testfile1 testfile
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

4. Transfer 1MB from offset 0 of HPSS file /home/bob/prod1 to offset 1048576 of local file testfile.

```
ftp> lfget -r 0 -l 1048576 -s 1048576 /home/bob/testfile1 testfile
```

```
... (Information returned by the PFTP Daemon)
```

```
ftp>
```

**3.2.10. Multiple Local file store - mlfput****Synopsis**

**mlfput** *local\_files*

**Description**

The **mlfput** is a performance optimized Parallel FTP Client command used to transfer multiple "globally available" files into HPSS using the "parallel" protocols. IF the file(s) is not available to the mover(s) machine (NOT "globally available") the transfer will fail because the mover(s) will NOT be able to locate the desired file. The difference between mlfput and mpput is that the mover(s) involved in the transfer, will not use the network to move the data providing improved performance. The mover(s) machine MUST be correctly configured. The file: /var/hpss/etc/hpss\_mvr\_localfilepath.conf MUST exist on each "local file" aware mover machine and MUST contain entries specifying which directories are eligible for "local file" transport.

The **mlfput** command expands the files specified in the *local\_files* parameter at the local host and copies the indicated files to HPSS. The **mlfput** command functions just like the standard ftp **mput** command.

**Parameters**

*local\_files*                      Identification of the files to transfer on the local machine.

**Return strings**

Output shows the amount of data transferred and any error conditions.

**Error conditions**

Connection Failures:    data transfer connection malfunction.

Network Failures:        data transfer malfunction.

Allocation Failures:    no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error codes are:

-5 - an I/O error occurred.

-28 - no space remaining in the associated storage class.

**See also**

RFC-0959.

**Notes**

none.

**Examples:**

1. Transfer all "globally available" files in the current directory to the user's HPSS home directory.

```
ftp> cd ~  
... (Information returned by the PFTP Daemon)  
ftp> prompt (<== Toggles File Prompting)  
...  
ftp> mlfput *  
... (Information returned by the PFTP Daemon)  
ftp>
```

2. Transfer all "globally available" files which begin with test in directory /usr/bob to the user's PSS home directory.

```
ftp> cd ~  
... (Information returned by the PFTP Daemon)  
ftp> mlfput /usr/bob/test*  
... (Information returned by the PFTP Daemon)  
ftp>
```

**3.2.11. Multiple Local file retrieval - mlfget****Synopsis**

**mlfget** *remote\_files*

**Description**

The **mlfget** is a performance optimized Parallel FTP Client command used to transfer an HPSS file into a "globally available" file using the "parallel" protocols. IF the current working directory or the specified directory is not available to the mover(s) machine (NOT "globally available") the transfer will fail because the mover(s) will NOT be able to locate the desired file. The difference between **mlfget** and **mpget** is that the mover(s) involved in the transfer, will not use the network to move the data providing improved performance. The mover(s) machine MUST be correctly configured. The file: /var/hpss/etc/ hpss\_mvr\_localfilepath.conf MUST exist on each "local file" aware mover machine and MUST contain entries specifying which directories are eligible for "local file" transport.

The **mlfget** command expands the *remote\_files* parameter at the remote (HPSS) host and copies the indicated HPSS files to the current directory on the local host. The **mlfget** command functions just like the standard ftp **mget** command.

**Parameters**

*remote\_files*                      Identification of the files to transfer from the remote (HPSS) host.

**Return strings**

Output shows the amount of data transferred and any error conditions.

**Error conditions**

Connection Failures:      data transfer connection malfunction.

Network Failures:         data transfer malfunction.

Allocation Failures:      no space on remote machine for file.

Error codes may also be returned from HPSS. The most common error code is:

-5 - an I/O error occurred.

**See also**

RFC-0959.

**Notes**

none.

**Examples**

1. Transfer all files in HPSS directory /home/bob to the "globally available" current working directory

```
ftp> mlfget /home/bob/*
```

```
...          (Information returned by the PFTP Daemon)
```

```
ftp>
```

2. Transfer all HPSS files which begin with test in directory /home/bob to the "globally available" current working directory.

```
ftp> mlfget /home/bob/test*
```

```
...          (Information returned by the PFTP Daemon)
```

```
ftp>
```

**3.2.12. Specify TCP socket based transfers - psocket**

**Synopsis**

**psocket**

**Description**

The **psocket** command is used to specify to the FTP client code that any parallel transfers are now to be done using connection based sockets (TCP).

**Parameters**

none.

**Return strings**

"Parallel transfers will now go over sockets."

**Error conditions**

none.

**See also**

RFC-0959.

**Notes**

**psocket** is the default.

### 3.2.13. Specify transfer stripe width - setpwidth

#### Synopsis

```
setpwidth stripe_width
```

#### Description

The **setpwidth** command is used to specify the size of the client side stripe to the FTP client code.

#### Parameters

*stripe\_width* The width of the PFTP client-side stripe. The width can have a value of 1 through 16. The default width is 1. The stripe width from the PFTP client perspective is the number of client processes spawned to handle the data transfers. Stripe width from the server perspective is the number of volumes the file is striped across.

A general guideline would be to set *stripe\_width* to an even divisor of the number of volumes the file is striped across. For example, if the Class of Service for a file were set up for a 4-way stripe, suggested values for *stripe\_width* might be 2 or 4.

If the stripe width of the file is unknown, consult your HPSS administrator or follow the following steps to determine the stripe width.

1. Enter the **lshpss -cos** command to list Class of Service information. From the entry with the *CID* value equal to your Class of Service ID, locate the *HID* (hierarchy ID) field.
2. Enter the **lshpss -hier** command to list hierarchy information. From the entry matching the *HID* value above, locate the *StorageClassID* field.
3. Enter the **lshpss -sc** command to list the storage class information. From the entry matching the *StorageClassID* value above, locate the *W* field. This is the stripe width for the storage class of the file.

#### Return strings

"Parallel stripe width set to [stripe width]."

#### Error conditions

"Bad width value [stripe width]."

#### See also

RFC-0959.

#### Notes

none.

### Example

1. Set the stripe width to 4.  
**ftp> setpwidth 4**

**3.2.14. Specify transfer block size - setpblocksize****Synopsis**

```
setpblocksize block_size
```

**Description**

The **setpblocksize** command is used to specify the block size to be used for parallel transfers. The *block\_size* may be specified using a decimal and magnitude representation string. See Section 2.1.10 for use of this notation. The maximum blocksize is 16mb.

**Parameters**

*block\_size*                      The number of bytes to be transferred to each element of the stripe before data is sent to the next element. The current allowable transfer sizes range from 1 through 16MB. The default block size is 256KB

A general guideline would be to set *block\_size* to the virtual volume block size. Consult your HPSS administrator or follow the following steps to determine the virtual volume blocksize.

1.        Enter the **lshpss -cos** command to list Class of Service information. From the entry with the *CID* value equal to your Class of Service ID, locate the *HID* (hierarchy ID) field.
2.        Enter the **lshpss -hier** command to list hierarchy information. From the entry matching the *HID* value above, locate the *StorageClassID* field.
3.        Enter the **lshpss -sc** command to list the storage class information. From the entry matching the *StorageClassID* value above, locate the *VVBlk* field. This is the virtual volume block size.

**Return strings**

"Parallel block size set to [block size]."

**Error conditions**

"Bad block size value [block size]."

**See also**

RFC-0959.

**Notes**

none.

**Example**

1. Set the transfer block size to 8 MB.

```
ftp> setpblocksize 8388608
```

or

```
ftp> setpblocksize 8MB
```

### 3.2.15. Multinode Enable/Disable - multinode

#### Synopsis

**multinode**

#### Description

The **multinode** command is used to enable/disable the "desire" to perform a parallel file transfer using multiple nodes. When multinode is enabled, the **pftp\_client** will process the multinode configuration file. If the process cannot obtain a single node to perform the parallel transfer, then the transfer will occur using non-multinode parallel method.

#### Parameters

None.

#### Return strings

"Processing the multinode list, please wait....."

"Multinode is on."

or

"Multinode is off."

#### Error conditions

"Configuration File I/O Problems: Without nodes, files cannot be transferred using the multiple node capability."

#### See also

None

### 3.2.16. Autoparallel Enable/Disable - autoparallel

#### Synopsis

**autoparallel**

#### Description

The **autoparallel** command is used to enable/disable the automatic mapping of non-parallel commands to parallel commands; e.g., `get ==>pget`. In autoparallel mode (enabled), transfers involving files smaller than the "Auto Parallel Size" specification in the HPSS.conf will NOT be auto-mapped.

#### Parameters

None.

#### Return string

Automatic Substitution of Parallel Commands Disabled

Daemon supports Parallel Features - Auto-Parallel Substitution Enabled

#### Error conditions

?Invalid command

#### See also

HPSS.conf(7)

#### Examples

```
ftp> autop
```

Automatic Substitution of Parallel Commands Disabled

or

Daemon supports Parallel Features - Auto-Parallel Substitution Enabled

**3.2.17. Get Current Protocol Mode - getprot**

**Synopsis**

getprot

**Description**

Display the current Parallel protocol mode.

**Parameters**

None.

**Return strings**

Current Parallel Protocol is PDATA and MOVER to MOVER

Current Parallel Protocol is PDATA ONLY

**Error conditions**

?Invalid command

==> Older Client?

**See also**

None.

**Examples**

```
ftp> getprot
```

Current Parallel Protocol is PDATA and MOVER to MOVER

Current Parallel Protocol is PDATA ONLY

**3.2.18. 3.2.19. Get Tuning Parameters - gettuningparms**

**Synopsis**

`gettun hostname/IP Addr`

**Description**

Display the (transfer) parameters between the Client and other hosts (default is the PFTP Daemon host.)

**Parameters**

None.

**Return strings**

See Example below.

Error conditions Warning Preceding without HPSS.conf (-2) (Observed at Login Time)

HPSS.conf parsing errors.

**See also**

None.

**Example**

```
ftp> gettun
```

```
Effective Tuning parameters from saux22 to sair031.sandia.gov
```

```
Using PDATA_AND_MOVER protocol
```

```
Using 4.1 Protocol
```

```
Parallel Transfer Size = 2147483647
```

```
Transfer Buffer Size = 16777216
```

```
Parallel Block Size = 262144
```

```
Parallel Network Width = 1
```

```
No Interfaces Found (ret_code = -2)
```

```
Using Default Interface for 1 stripes(s)
```

```
Multinode is disabled
```

or

Multinode Enabled:

Processing the multinode list, please wait.....

Using 1 remote node(s) from the following:

Control Interface ==> Data Interface:

water ==> water

Using Network Options

Using "Default" destination characteristics

PdataSockBufSize = 1048576 based on user input

recv\_socksize = send\_socksize = 1048576 based on PdataSockBufSize

Writesize = 524288

or

recv\_socksize = 262144 based on Network Options

send\_socksize = 262144 based on Network Options

PdataSockBufSize = 262144 based on send\_socksize

RFC1323 is turned on

TCPNoDelay is turned on

*NOTE: Parallel Pipes are NOT supported regardless of the following:*

PipeFileSize TOO Large reset to 2147483647

Pipe Files NOT supported on this machine - Open Failed 2

or

Pipe Files are supported on this machine

Pipe File = /copyvol/.pftp\_pipes26404

Pipe File Size = 1073741824

**3.2.19. Set the PDATA\_ONLY protocol - pdata**

**Synopsis**

pdata

**Description**

Explicitly request the PDATA\_ONLY protocol.

**Parameters**

None.

**Return strings**

\*\*\*\* NOTE: Protocol set to PDATA\_ONLY \*\*\*\* (At logon time)

215 Parallel protocol is PDATA\_ONLY

**Error conditions**

?Invalid command ==> Older Client?

**See also**

None.

**Examples**

1. Set protocol to PDATA\_ONLY (Failure)

ftp> **pdata**

Server does NOT support command ==> Older Server?

?Invalid command ==> Older Client?

ftp>

### 3.2.20. Override the Non-AutoParallel mode - **penable**

#### Synopsis

penable

#### Description

Explicitly specify parallel transfers. This is useful when transferring data to a Parallel FTP Daemon which is unaware of the PRLI command. This command is included for backward compatability only and may be discontinued at any time.

#### Parameters

None.

#### Return strings

None

#### Error conditions

?Invalid command

==> Older Client?

#### See also

None.

#### Example

1. Set parallel enable

```
ftp> penable
```

```
ftp> (No Response)
```

**3.2.21. Set the PDATA\_AND\_MOVER protocol - pmover**

**Synopsis**

pmover

**Description**

Explicitly specify parallel transfers to use the PDATA\_AND\_MOVER protocol (Original protocol) regardless of what is specified in the HPSS.conf file.

**Parameters**

None.

**Return strings**

215 Parallel protocol is PDATA\_AND\_MOVER

**Error conditions**

ftp> pmover

Server does NOT support command

==> Older Server?

?Invalid command

==> Older Client?

**See also**

HPSS.conf(7)

**Examples**

1. Set PDATA\_AND\_MOVER protocol

ftp> **pmover**

215 Parallel protocol is PDATA\_AND\_MOVER

ftp>

### 3.2.22. Set the Socket Buffer Size - setsockbufsize

#### Synopsis

setsock *SizeString*

#### Description

Set the desired socket buffer size. Useful when no HPSS.conf file exists or the client/mover combination is NOT in the HPSS.conf file. When entered without a *SizeString*, the command returns the Socket Buffer Size in effect.

#### Parameters

*SizeString*; e.g., "1MB"

#### Return strings

Socket Buffer Size = 1048576.

#### Error conditions

PdataSockBufSize reset equal or below sb\_max (1048576)

#### See also

None.

#### Example

1. Set Socket Buffer Size

```
ftp> setsock 4mb (Above system max)
```

```
PdataSockBufSize reset equal or below sb_max (1048576)
```

```
ftp>
```

2. Set Socket Buffer Size

```
ftp> setsock 512kb
```

```
ftp> (No Response)
```

3. Set Socket Buffer Size (no argument)

```
ftp> setsock
```

```
Socket Buffer Size = 524288.
```

```
ftp>
```

### 3.2.23. Set the Transfer Buffer Size - setxferbufsize

#### Synopsis

setxferbufsize *SizeString*

#### Description

Set the desired transfer buffer sizes. Useful when no HPSS.conf file exists or the client/daemon combination is NOT in the HPSS.conf file. When entered without a *SizeString*, the command returns the Transfer Buffer Size in effect.

#### Parameters

*SizeString*, e.g., "4MB"

#### Return strings

PdataBufferSize = 4194304

#### Error conditions

?Invalid command

==> Old Client?

#### See also

None.

#### Example

1. Set Transfer Buffer Size

```
ftp> setxfer 40MB (Max is 32MB)
```

```
ftp> (No Response)
```

2. Display effective Transfer Buffer Size (no argument)

```
ftp> setxfer
```

```
Socket Buffer Size = 33554432.
```

```
ftp>
```

---

## Chapter 4. User Utilities

HPSS provides a set of utilities for administrators and users. The majority of the HPSS utilities are for administrators, and are defined in the HPSS Administration Guide. Those utilities applicable to users are documented in this chapter. There are also man pages for these utilities.

Note: The user must log in to DCE prior to using the utilities.

### 4.1. Utilities

The user utilities defined in the chapter are:

**hacl**

**lshpss**

#### 4.1.1. HPSS ACL Editor - hacl

##### Synopsis

```
hacl [-x] [-v] [command ... ]
```

##### Description

The hacl utility is used to manage access control lists (ACLs) on HPSS files. In particular, hacl can replace an ACL with a different one, add new ACL entries or update existing ones, delete selected ACL entries, clear out an entire ACL, and display ACLs. To make it easier to change ACLs, hacl can also copy an ACL from an existing file to another file.

If a command is supplied on the execute line, then the command is executed immediately and control is returned to the shell. Otherwise hacl enters interactive mode. In this mode, the user can change working directories, look at the files in the current directory, try out acl changes, and see the results of the change. Interactive mode may also be used to read commands from a script.

Access control lists are made up of individual ACL entries separated by "white space". White space characters include commas, semicolons, and newlines, as well as spaces and tabs.

Each ACL entry takes the form "<type>:<key>:<perms>", where:

<type>                    the ACL entry type (e.g., "user\_obj")

<key>                    principal and/or cell name

<perms>                  the permissions (e.g., "rwx")

The <type> field can be any of the standard DCE ACL entry types, including user\_obj, group\_obj, other\_obj, user, group, foreign\_user, foreign\_group, foreign\_other, any\_other, mask\_obj, unauthenticated, user\_obj\_delegate, group\_obj\_delegate, other\_obj\_delegate, user\_delegate, group\_delegate, foreign\_user\_delegate, foreign\_group\_delegate, foreign\_other\_delete, and any\_other\_delegate.

The <key> field takes the form /<cell>/<principal>. The <cell> or <principal> may or may not be required, depending on the ACL entry type. The <cell> can be specified with either a cell name (e.g., /.../hpss.ca.sandia.gov) or a cell id (e.g., /87654321).

Similarly, users can be specified either by name (e.g., hiliary) or by uid (e.g., 20021), and groups can be specified by name or gid. Normally the user will use names rather than ids, but ids are provided to deal with situations where the cell, principal, and/or group have been removed from the security registry.

The <perms> field may or may not be required, depending on the command. When required, the field is made up the characters "rwxcid", signifying respectively read, write, execute, control, insert, and delete. To specify that a permission is not wanted, use a hyphen (-) or just omit the character corresponding to that permission.

Many commands can optionally act on the initial container or initial object ACL of a directory. To specify one of these, use the -ic or -io flag on the command line. For example, to clear the initial container ACL on mydirectory:

```
$ hacl clear -ic -f mydirectory
```

The -ic and -io flags are mutually exclusive. If neither is present on the command line, then the object ACL itself will be processed.

Many commands require that a list of ACL entries be entered. The list can be specified in one of two ways using the -a or -A flags. The flags are mutually exclusive. In the command descriptions below, these flags are documented using the following notation:

```
<command> ... [-a <entries>] [-A <filea>] ...
```

If the -a flag is used, then <entries> is an explicit list of ACL entries to be processed. If the -A flag is used, then <filea> is the name of Unix file that contains the ACL entries. The format of the file is flexible, but it is recommended that each ACL entry appear on a separate line.

Many commands also require the user to specify the list of files that are to be processed. The list can be specified in one of three ways using the -B, -f, and -F flags. The flags are mutually exclusive. In the command descriptions, this notation is used:

```
<command> ... [-B <fileb>] [-F <filef>] [-f <files>]
```

If the -f flag is used, it must be the last flag on the command line. In this case, <files> is a list of files to be processed. The list may contain files whose names begin with a hyphen. With the exception of the "ls" command, the list may not contain wildcards or escape characters.

If the -F flag is used, then <filef> is the name of a Unix file that contains the names of the HPSS files to be processed. The file must be formatted with one file to a line. Leading blank characters are ignored, but embedded and trailing blanks are considered to be part of the file name. Asterisks (\*), question marks (?), and escapes (\) are not translated but rather are treated as part of the file name.

In addition a bulk file processing option may be available in some versions of hacl. This option, selected by the -B flag, allows hacl to change ACLs more efficiently. To use the option, a Unix file must first be prepared containing the HPSS nameserver handle and the name of each file that is to be processed. To prepare the Unix file, use hacl's "ls -h" command. Once the file has been prepared, use hacl's -B flag, naming the Unix file as <fileb>. One of the examples below shows how to do this in more detail.

To designate input from stdin, use a hyphen (-) for the <filea>, <fileb>, or <filef> parameters. Needless to say, only one of these parameters can be set to stdin at a time. Moreover, the hyphen cannot be used in interactive mode.

The "ls" is the only hacl command that accepts wildcards and escape characters. The asterisk (\*) and question mark (?) have the same interpretation as they do in most Unix shells. The backslash (\) is an escape character that can be used to represent troublesome characters. The following escape sequences are recognized:

\t	the tab character
\'	the apostrophe (')
\"	the double quote (")
\(blank)	the blank ( ) character
\nnn	the ASCII character given by 1-3 octal digits
\\	the backslash (\) character itself
\(other)	unchanged

Escape sequences are translated before wildcards are evaluated, which has the practical effect that there is no simple way to specify a file name that contains an asterisk or question mark.

Note that file names can either be specified as absolute or relative path names. When hacl starts, it determines the user's default working directory from HPSS. However, if the environment variable **HACL\_DEFAULT\_DIR** is set, then its value will determine the default directory. The directory can be changed by using hacl's "cd" command.

Some commands support a -q flag. If the flag is missing, then information is printed out for every file that is processed. For example:

```
$ hacl copy -m model -f file.01 file.02  
  
file.01 OK  
  
file.02 Cannot change ACL. (No such name)
```

If the -q flag is present, only the commands that fail will be reported. This makes it easier to see errors. For example:

```
$ hacl copy -q -m model -f file.01 file.02  
  
file.02 Cannot change ACL. (No such name)
```

By default, when hacl encounters an error, it continues running, processing other files and hacl commands until a quit or exit command is seen. However, this can be dangerous if hacl is being run from a script. For example, if a "cd" command fails, hacl might end up processing files in the wrong directory. To prevent this from happening, always use the -x flag when running from a script. The -v flag should also be used so that hacl's messages can be related to the commands that generated them.

### Parameters

command	Optional one-liner command and parameters.
-x	Terminate commands as soon as an error occurs.
-v	Echo hacl commands to stdout.

### Command Summary

acl	Description of the syntax for access control lists
cd	Change the working directory
clear	Clear the entire ACL for one or more files
copy	Copy the ACL from one file to one or more other files
echo	Echo the arguments to standard output
exit	Leave the program
help	Get help on how to use hacl
ls	List the files in a directory
purge	Purge unused ACL perms to account for mask object
pwd	Print the current working directory
remove	Remove selected ACL entries from one or more files
replace	Replace entire ACL with new ACL for one or more files
quit	Leave the program
show	Show the ACL for one or more files
update	Update selected ACL entries for one or more files
usage	Describe hacl's command line

### Command Details

**acl**

Description of the syntax for access control lists

This is not a command per se, but rather a topic for which help is available. Typing the command "help acl" results in a description of the syntax for ACLs.

### **cd [<dir>]**

Change the working directory

<dir>                      New working directory

If <dir> is missing, this command changes the current working directory to be the user's default working directory.

### **clear [-ic] [-io] [-q] [-B <fileb>] [-F <filef>] [[-f] <files>]**

Clear the entire ACL for one or more files

-ic	the ACL is an initial container ACL
-io	the ACL is an initial object ACL
-q	don't report successes
-B	bulk ACL change for files listed in <fileb>
<fileb>	a Unix text file listing file names and NS handles
-F	file names are determined by reading <filef>
<filef>	a Unix text file listing file names
-f	file names explicitly listed in <files>
<files>	an explicit list of files

This command deletes all ACL entries except the user\_obj, group\_obj, other\_obj entries.

See the discussion section for a description of the flags.

### **copy [-ic] [-io] [-q] -m <filem> [-B <fileb>] [-F <filef>] [[-f] <files>]**

Copy the ACL from one file to one or more other files

-ic	the ACL is an initial container ACL
-io	the ACL is an initial object ACL
-q	don't report successes
-m	change ACL to that given by "model" file <filem>
<filem>	an HPSS file whose ACL will be copied

<b>-B</b>	bulk ACL change for files listed in <fileb>
<fileb>	a Unix text file listing file names and NS handles
<b>-F</b>	file names are determined by reading <filef>
<filef>	a Unix text file listing file names
<b>-f</b>	file names explicitly listed in <files>
<files>	an explicit list of files

The **-m** flag is used to specify the name of the HPSS file whose ACL is to be copied. This flag is required. See the discussion section for a description of the other flags.

### **echo [<text>]**

Echo the arguments to standard output

<text>	the message text to be echoed
--------	-------------------------------

### **exit**

Leave the program

### **help [-v] [<topic>]**

Get help on how to use hacl

<b>-v</b>	print more details
<topic>	topic for which help is desired

If <topic> is omitted, print a list of topics for which help is available.

### **ls [-a] [-h] [-l] [-f] [<files>]**

List the files in a directory

<b>-a</b>	list files whose names begin with a dot (.)
<b>-h</b>	list file handles needed by the bulk ACL commands
<b>-l</b>	list owner's cell name, user name, and group name
<b>-f</b>	list the files named in <files>

The **-f** flag must be last on the command line. The <files> argument can contain wildcards and escape characters. If one of the files named is a directory, the directory itself will be listed rather than the files in that directory. If <files> is omitted, then all of the files in the current working directory will be listed. The "ls" command is the only hacl command that accepts wildcards. To use wildcards on other commands, the output of the "ls" command must be directed into a file or a pipe and then read back into hacl using the **-F** flag. For example, this will show the ACLs for all files whose names end in ".dat":

**\$ hacl ls -f '\*.dat' | hacl show -F -**

If the `-h` flag is used, the output of the "ls" command will be in the right format to use as input to one of the bulk ACL operations. For example, the following command copies the ACL in `myfile.dat` to all the files in the current working directory:

**\$ hacl ls -h -f '\*' | hacl copy -m myfile.dat -B -**

**purge [-ic] [-io] [-q] [-B <fileb>] [-F <filef>] [[-f] <files>]**

Purge unused ACL permissions to account for the mask object

<code>-ic</code>	the ACL is an initial container ACL
<code>-io</code>	the ACL is an initial object ACL
<code>-q</code>	don't report successes
<code>-B</code>	bulk ACL change for files listed in <fileb>
<fileb>	a Unix text file listing file names and NS handles
<code>-F</code>	file names are determined by reading <filef>
<filef>	a Unix text file listing file names
<code>-f</code>	file names explicitly listed in <files>
<files>	an explicit list of files

See the discussion section for a description of the flags.

## **pwd**

Print the current working directory

**remove [-ic] [-io] [-q] [-a <entries>] [-A <filea>] [-B <fileb>] [-F <filef>] [-f <files>]**

Remove selected ACL entries from one or more files

<code>-ic</code>	the ACL is an initial container ACL
<code>-io</code>	the ACL is an initial object ACL
<code>-q</code>	don't report successes
<code>-a</code>	ACL entries are explicitly listed
<entries>	explicit list of ACL entries
<code>-A</code>	ACL is determined by reading a file
<filea>	Unix text file listing the new ACL entries

<b>-B</b>	bulk ACL change for files listed in <fileb>
<fileb>	a Unix text file listing file names and NS handles
<b>-F</b>	file names are determined by reading <filef>
<filef>	a Unix text file listing file names
<b>-f</b>	file names explicitly listed in <files>
<files>	an explicit list of files

See the discussion section for a description of the flags.

**replace [-ic] [-io] [-q] [-a <entries>] [-A <filea>] -B <fileb>] [-F <filef>] [-f <files>]**

Replace entire ACL with new ACL for one or more files

<b>-ic</b>	the ACL is an initial container ACL
<b>-io</b>	the ACL is an initial object ACL
<b>-q</b>	don't report successes
<b>-a</b>	ACL entries are explicitly listed
<entries>	explicit list of ACL entries
<b>-A</b>	ACL is determined by reading a file
<filea>	Unix text file listing the new ACL entries
<b>-B</b>	bulk ACL change for files listed in <fileb>
<fileb>	a Unix text file listing file names and NS handles
<b>-F</b>	file names are determined by reading <filef>
<filef>	a Unix text file listing file names
<b>-f</b>	file names explicitly listed in <files>
<files>	an explicit list of files

See the discussion section for a description of the flags.

### **quit**

Leave the program

**show [-e] [-ic] [-io] [-B <fileb>] [-F <filef>] [[-f] <files>]**

Show the ACL for one or more files

-e	show effective permissions considering mask object
-ic	the ACL is an initial container ACL
-io	the ACL is an initial object ACL
-B	bulk ACL change for files listed in <fileb>
<fileb>	a Unix text file listing file names and NS handles
-F	file names are determined by reading <filef>
<filef>	a Unix text file listing file names
-f	file names explicitly listed in <files>
<files>	an explicit list of files

See the description section for a description of the flags.

**update [-ic] [-io] [-m <type>] [-q] [-a <entries>] [-A <filea>] [-B <fileb>] [-F <filef>] [-f <files>]**

Update selected ACL entries for one or more files

-ic	the ACL is an initial container ACL
-io	the ACL is an initial object ACL
-q	don't report successes
-m	adjust mask object according to <type>
<type>	either "calc" or "nocalc"
-a	ACL entries are explicitly listed
<entries>	explicit list of ACL entries
-A	ACL is determined by reading a file
<filea>	Unix text file listing the new ACL entries
-B	bulk ACL change for files listed in <fileb>
<fileb>	a Unix text file listing file names and NS handles
-F	file names are determined by reading <filef>
<filef>	a Unix text file listing file names
-f	file names explicitly listed in <files>
<files>	an explicit list of files

See the discussion section for a description of the flags.

This command can be used to add new entries to an ACL as well as to change existing entries.

### usage

Describe hacl's command line usage

### Examples

To show the ACL on a file:

```
$ hacl show myfile.dat
```

To show the effective ACL on a file:

```
$ hacl show -e myfile.dat
```

To show the initial container ACL on a directory:

```
$ hacl show -ic mydir
```

To copy an ACL from a model file to several other files:

```
$ hacl copy -m model.file -f file.00 file.01 file.02
```

To copy an ACL to a list of files specified using wildcards:

```
$ hacl ls '*.dat' | hacl copy -m model.file -F -
```

To update a single ACL entry on a single file:

```
$ hacl update user:hilary:rwX file.00
```

To work with several ACL entries and files:

```
$ hacl update -a user:joe:rwX user:ann:cid -f f1 f2 f3
```

To run hacl interactively and work with mydir/file.00:

```
$ hacl  
> show mydir/file.00  
> cd mydir  
> pwd  
> ls  
> update -a user:hilary:rwX -f file.00  
> show file.00
```

To establish a default working directory (using ksh):

```
$ export HACL_DEFAULT_DIR=/home/hilary/mydir  
$ hacl  
> show file.00
```

To use an ACL specified in a Unix file:

```
$ cat myacls  
user:joe:rwx  
user:ann:cid  
$ hacl update -A myacls -f f1 f2 f3
```

To modify the files listed in a Unix file:

```
$ cat myfiles  
f1  
f2  
f3  
$ hacl update -a user:joe:rwx user:ann:cid -F myfiles
```

To run hacl using a script:

```
$ cat myscript  
update -A myacls -f f1 f2 f3  
update -a user:joe:rwx user:ann:cid -F myfiles  
$ hacl -v -x < myscript
```

To generate and use the info needed by a bulk ACL command

```
$ hacl ls -h '*.01' > mybulk  
$ cat mybulk  
000002370000023700000000af4a81017ec823feee7511d1ab4708005a4726ef acl.01  
0000023e0000023e00000000273381017ec823feee7511d1ab4708005a4726ef ftp.01  
$ hacl update -A myacls -B mybulk
```

See Also

None.

### **4.1.2. List information about HPSS - lshpss**

#### **Synopsis**

**lshpss** [options ...]

#### **Description**

**lshpss** displays HPSS resources, such as Class of Service, Hierarchy, and Storage Class. Before running this script, you must be authorized to access the SFS files. This is accomplished by entering **dce\_login** for an authorized DCE ID.

#### **Parameters**

<b>-glob</b>	Show Global Configuration Data
<b>-subsys</b>	Show Storage Subsystems
<b>-cos</b>	Show Class of Service list
<b>-hier</b>	Show Hierarchy list
<b>-sc</b>	Show Storage Class list
<b>-migp</b>	Show Migration Policies
<b>-purgep</b>	Show Purge Policies
<b>-vol</b>	Show Physical Volumes
<b>-dev</b>	Show Mover Devices
<b>-drv</b>	Show PVL Drives
<b>-svr</b>	Show HPSS Servers
<b>-mvr</b>	Show HPSS Movers
<b>-acct</b>	Show Accounting Policies
<b>-logp</b>	Show Log Policies
<b>-locp</b>	Show Location Policies
<b>-ffam</b>	Show file families
<b>-bfs</b>	Show Bitfile Servers
<b>-ns</b>	Show Name Servers

<b>-pvr</b>	Show PVRs
<b>-logd</b>	Show Log Daemons
<b>-logc</b>	Show Log Clients
<b>-nfsd</b>	Show NFS Daemons
<b>-listmeta</b>	List metadata
<b>-all</b>	Display all of the above
<b>-dumpmeta</b>	Dump all metadata for HPSS to separate Unix files
<b>-dump <i>file</i></b>	Dump one metadata file to a Unix file of the same name
<b>-sdt</b>	Go into an sdt shell
<b>-g <i>globalconfig</i></b>	Indicate CDS pathname of HPSS global configuration metadata file to use. This value can also be supplied by setting the <b>HPSS_CONFIG_GLOBAL</b> environment variable.
<b>-h</b>	Show this help message

**See also**

None.

**Notes**

The option of interest to most users is the **-cos** option. This option allows the user to view a list of all defined Classes of Service.

In previous versions of HPSS, it was necessary to log in to DCE before running this command. In this version, the utility will first check to see if the user is already authenticated with DCE. If so, the utility will use the existing DCE context. If the user is not authenticated, the utility will automatically attempt to log in to DCE as a user with the privileges necessary to perform the necessary metadata operations.

**Example**

The following lists the Classes of Service, hierarchies, and storage classes:

**lshpss -cos -hier -sc**

Sample output from this example is shown below:

```
Logging into DCE as "hpss_ssm" with keytab "/krb5/hpss.keytabs"...
Using default global config file
Using global config file: ../encina/sfs/hpss/globalconfig

      - Class of Service List -

COS Hier      Optim      Xfer Avg
ID  ID  Name      Access      Rate Lat
ID  ID  Name      Size      Min / Max      kB/s (s) SC Flags
```

```

-----
 1  1 larry: very small files (H1)      4MB          0/    17179869184 4096  0  O RWAM-
 2  2 larry: small files/2 copy (H2)   4MB        4194304/    16777216 4096  0  O RWAM-
 3  3 larry: small files (H3)          4MB        4194304/    16777216 4096  0  O RWAM-
 4  4 larry: medium files (H4)         4MB       16777216/    68719476736 4096  0  O RWAM-
 5  5 larry: large files (H5)          4MB       67108864/   214748364800 4096  0  O RWAM-
 6  6 larry: 1w tape files (H6)        4MB          0/    107374182400 4096  0  O RWAM-
 7  7 larry: 2wt tape files (H7)       4MB          0/    107374182400 4096  0  O RWAM-
 8  3 larry: small files_b (H3)        4MB          4/     16777216 4096  0  A RWA--
-----

```

SC=Stage Code (N=none, O=on open, A=async, B=background)

Flags: RWAMF

R=Enable read                    W=Enable write                    A=Enable append  
M=Enforce max file size        F=Force selection

- Hierarchy List -

```

-----
Hier          #
ID Description  Lev Storage Class IDs
-----
 1 larry: 1wd -> 1wt (H1)          2 1 --> 101
 2 larry: 1wd -> 1wt -> 1wt (H2)   3 2 --> 102 --> 103
 3 larry: 1wd->1wd->1wt->1wt (H3)  4 3 --> 4 --> 104 --> 105
 4 larry: 2wd -> 1wt (H4)          2 5 --> 106
 5 larry: 4wd -> 1wt (H5)          2 6 --> 107
 6 larry: 1wt -> 1wt (H6)          2 108 --> 109
 7 larry: 2wt (H7)                 1 110
-----

```

- Disk Storage Class List -

SC	Media	Xfer Rate	SSEG Size	Est Avg # PV	Media Str	Thresh Block	Mig Pol	Prg Pol
ID Name	Type	(kB/s)	min/max	SSegs Size	VBBS Wid Size	(% use) Wrn/Crt	ID	ID
1	larry: 1wd (H1-L1)	Default	3072 256K/ 1MB	4 4GB	256K 1	4K 80/ 90	1	1
2	larry: 1wd (H2-L1)	Default	3072 2MB/ 4MB	4 4GB	1MB 1	4K 80/ 90	2	1
3	larry: 1wd (H3-L1)	Default	3072 2MB/ 4MB	4 4GB	1MB 1	4K 80/ 90	0	0
4	larry: 1wd (H3-L2)	Default	3072 2MB/ 4MB	4 4GB	1MB 1	4K 80/ 90	1	1
5	larry: 2wd (H4-L1)	Default	12288 8MB/ 16MB	4 4GB	1MB 2	4K 80/ 90	1	1
6	larry: 4wd (H5-L1)	Default	49152 32MB/256MB	4 4GB	1MB 4	4K 80/ 90	1	1

VBBS=Virtual volume block size

- Tape Storage Class List -

SC	Media	Media Block	Est PV	Str	Xfer Rate	Threshold	Max	Mig	Prg
ID Name	Type	Size	Size	Wid VBBS	(kB/s) BBTM	(volumes) Warn/Crit	VVs to Write	Pol ID	Pol ID
101	larry: 1wt 3590E (H1-L2)	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	0	0
102	larry: 1wt 3590E (H2-L2)	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	0	0
103	larry: 1wt 3590E (H2-L3/2nd cp	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	0	0
104	larry: 1wt 3590E (H3-L3)	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	1	0
105	larry: 1wt 3590E (H3-L4)	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	0	0
106	larry: 1wt 3590E (H4-L2)	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	0	0
107	larry: 1wt 3590E (H5-L2)	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	0	0
108	larry: 1wt 3590E (H6-L1)	3590	256K 40GB	1 1MB	14336	280 2/ 1	1	1	0
109	larry: 1wt 3590E (H6-L2)	3590	256K 40GB	1 1MB	14336	280 2/ 1	2	0	0
110	larry: 2wt 3590E (H7-L1)	3590	256K 40GB	2 1MB	57344	560 2/ 1	2	0	0

BBTM=Blocks between tape marks

VBBS=Virtual volume block size

## Appendix A - Acronyms

ACL	Access Control List
ACLS	Automated Cartridge System Library Software (Science Technology Corporation)
AIX	Advanced Interactive Executive
API	Application Program Interface
CDS	Cell Directory Server
COS	Class of Service
DCE	Distributed Computing Environment
DFS	Distributed File System
EFS	External File System
FTP	File Transfer Protocol
GSS	Generic Parallel File System
gid	Group Identifier
GPFS	IBM General Parallel File System
GSS	Generic Security Service
HIPPI	High Performance Parallel Interface
HPSS	High Performance Storage System
IBM	International Business Machines Corporation
LaRC	Langley Research Center
LANL	Los Alamos National Laboratory
LLNL	Lawrence Livermore National Laboratory
NASA	National Aeronautics and Space Administration
ORNL	Oak Ridge National Laboratory
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input / Output
IP	Internet Protocol
NFS	Network File System
OSF	Open Software Foundation
PFTP	Parallel File Transfer Protocol
RISC	Reduced Instruction Set Computer
SFS	Structured File Server
SNL	Sandia National Laboratories
SP	Scalable Processor
TCP	Transmission Control Protocol
uid	User Identifier
VV	Virtual Volume



## Appendix B - References

1. *File Transfer Protocol, RFC-0959*, October 1985.
2. *HPSS Error Messages Manual*, August 2001.
3. *HPSS Programmer's Guide Reference, Volume 1*, August 2001.
4. *HPSS Programmer's Guide Reference, Volume 2*, August 2001.
6. *HPSS Installation Guide*, August 2001.
7. *HPSS Management Guide*, August 2001.
8. *Installing, Managing, and Using the IBM AIX Parallel I/O File System*, Document Number H34-6065-00.
9. *Network File System Specification, RFC-1094*, DDN Network Information Center, SRI International, Menlo Park, Ca.
10. *OSF DCE User's Guide and Reference*, Prentice Hall, Englewood Cliffs, N. J.



---

## Index

### A

ACL editor, 4-1

allocating space for files, 2-6

### B

block size

    specifying, 3-31

### C

chgid, 2-2

chgrp, 2-2

chmod, 2-3

chown, 2-4

chuid, 2-4

class of service (COS)

    concept, 1-5

class of servicee (COS)

    specifying 2-2

### D

DCE user accounts, 1-9

Distributed File System (DFS), 1-3

### F

file family, 1-6

File Transfer Protocol (FTP), 1-1, 2-1

filesets

    archived, 1-3

    mirrored 1-3

### G

group

    changing by ID, 2-2

    changing by name, 2-2

### H

hacl, 4-1

HPSS.Conf configuration file, 3-4

### I

interfaces

    usage considerations, 1-7

    user, 1-1

### K

krb5\_gss\_pftp\_client, 3-1

### L

lfappend, 3-18

lfget, 3-22

---

lftp, 3-20  
list directory extensions, 2-7  
list information about HPSS, 4-12  
local file append, 3-18  
local file retrieval, 3-22  
local file store, 3-20  
ls -lh, 2-7  
lshpss, 4-12  
M  
mlfget, 3-26  
mlfput, 3-24  
mpget, 3-16  
mpput, 3-12  
multinode  
    command, 3-33  
    enable / disable, 3-33  
multiple local file retrieval, 3-26  
multiple local file store, 3-24  
N  
Network File System Version 3 (NFS V3), 1-2  
O  
owner  
    changing by ID, 2-4  
    changing by name, 2-4  
P  
pappend, 3-8  
parallel append, 3-8  
parallel file retrieval, 3-14, 3-16  
parallel file store, 3-10, 3-12  
Parallel File Transfer Protocol, 3-1  
Parallel FTP (PFTP), 1-1  
permissions  
    changing, 2-3  
pftp\_client, 3-1  
pget, 3-14  
pput, 3-10  
psocket, 3-28  
Q  
quote allo64, 2-6  
S  
setcos, 2-2  
setpblocksize, 3-31  
setpwidth, 3-29  
site commands, 2-1  
stage, 2-5  
staging a file, 2-5

---

storage class, 1-6  
storage concepts, 1-5  
storage hierarchy, 1-6  
stripe width  
    specifying, 3-29  
symbolic link  
    creating, 2-6  
symlink, 2-6  
T  
TCP socket based transfers  
    specifying, 3-28  
U  
user IDs, 1-8  
utilities, 1-4, 4-1  
W  
wait for file to be staged, 2-5  
wait, 2-5