HPSS User’s Guide

High Performance Storage System, version 10.2.0.0.0, 15 February 2023

HPSS User's Guide

High Performance Storage System, version 10.2.0.0.0, 15 February 2023

Table of Contents

List of Figures

List of Tables

About this book. The High Performance Storage System (HPSS) User’s Guide provides the necessary information for transferring files using HPSS. In particular, the following interfaces are described:

- File Transfer Protocol (FTP)
- Parallel FTP (PFTP)
- HPSS Virtual File System (VFS)

It is not the intent of this document to define the standard commands and subcommands provided by standard FTP and POSIX compliant file systems. Only interface extensions provided by HPSS are defined within the *HPSS User's Guide*.

Refer to the *HPSS Installation Guide* and the *HPSS Management Guide* for descriptions of the interfaces provided to HPSS administrators. Refer to the *HPSS Programmer's Reference* for programming interfaces provided to the end user. Refer to the *HPSS Error Messages Manual* for a list of all HPSS error and advisory messages which are output by the HPSS software.

The *HPSS User's Guide* is structured as follows:

- **Chapter 1: Overview** - 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)** - Defines the extensions to the standard FTP interface.
- **Chapter 3: Parallel File Transfer Protocol (PFTP)** - Defines the Parallel FTP (PFTP) interface.
- **Appendix A: Glossary of Terms and Acronyms**
- **Appendix B: References** - Lists documents cited in the text as well as other reference materials.
- **Appendix C: Developer Acknowledgments**

Typographic and keying conventions. This document uses the following typographic conventions:

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

```
% sample command
```

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

```
# This is a comment
```

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.

Chapter 1. Overview

The High Performance Storage System (HPSS) has been designed to meet the high end of archival storage system and data management requirements. These requirements have led to a scalable design which uses network attached storage devices to transfer data at rates up to multiple gigabytes per second into data stores of many petabytes.

Listed below are the user interfaces available for accessing data in the 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 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 PFTP Daemon supports access from any FTP client that conforms to RFC-0959. In addition, the **quote allo64** command is supported. Additional "site" commands are available as described in Section 3.2, "PFTP **site** (and **quote**) commands" below.

Proxy (third-party) transfers are not supported. Also, to maximize performance, the user should explicitly change the data transfer type to binary. ASCII transfers are very inefficient as the daemon must scan the entire file to determine how many carriage returns to insert so that it can compute the actual amount of date to transfer. It may also cause incorrect results if a file is stored as binary and retrieved as ASCII, or vice versa.

Refer to the *HPSS Installation Guide* for information on configuring PFTP.

1.1.2. Parallel FTP (PFTP)

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 presented 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 via TCP/IP and communicates directly with HPSS Movers to transfer data.

The following constraints are imposed by PFTP.

- Outbound "Pipes" may be configured and supported. Inbound "Pipes" (pgets) are *not* supported. The use of "Pipes" should be discouraged.
- Proxy (third-party) transfers are not supported.
- ASCII transfers are not supported over the parallel interface. Since extra characters are inserted in the stream by the ASCII translation, there is no way to resolve data placement in a parallel stream. Note that some standard FTP implementations default to ASCII or to "Auto". If either of these are 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.

1.1.3. API Tools

HPSS provides a set of basic API tools for use in HPSS. See the individual tool man pages for details. The purpose of these tools is to provide a convenient command-line tool suite into HPSS for administrators for ease of scripting and basic system use. These tools fill a niche in HPSS to provide a basic set of tools for interacting with HPSS out of the box.

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

The following tools are provided:

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

14.**hpssstat** - Stat an HPSS file

15.**hpsstouch** - Touch an HPSS file

1.1.4. HPSS Virtual File System (VFS) interface

Support for HPSS VFS has been deprecated in favor of an analogous application called HPSSFS-FUSE. Refer to HPSSFS-FUSE documentation for its usage.

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 (COS)

A 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. The desired COS is selected when the file is created. Underlying a COS is a storage hierarchy that describes a progression of storage media that may be used to store the file.

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. Contact your HPSS administrator to determine the COSs which have been defined and available for your use.

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.

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 five storage classes. Storage hierarchies and storage classes are not directly visible to the user, but are described in the following sections.

1.2.2. Storage class

An HPSS storage class is used to group various storage media together to provide them with uniform characteristics. The attributes associated with a storage class are both physical and logical. Physical storage media in HPSS are called physical volumes. Physical characteristics associated with physical volumes are the media type, block size, the estimated amount of storage space on volumes in this class, and how often to write tape marks on tape volumes.

Physical volumes are organized into logical virtual volumes. Virtual volumes can be striped by aggregating two or more physical volumes in the virtual volume. Some of the virtual volume 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 disk storage segment size. In addition, the storage class has attributes that associate it with a particular migration policy and purge policy to help in managing the usage of media in the storage class.

RAIT volume characteristics are described in storage classes. When a tape storage class is defined to have three or more physical volumes in a stripe, and a nonzero value for the ParityStripeWidth, then that storage class defines RAIT volumes. Virtual volumes created in such a storage class will have RAIT characteristics.

1.2.3. Storage hierarchy

An HPSS storage hierarchy consists of one to five storage classes in a fixed order. Files are moved up and down the storage hierarchy via stage and migrate operations, based upon storage policy, usage patterns, storage availability, and user request. For example, a storage hierarchy might consist of a fast disk storage class, followed by a fast data transfer and medium storage capacity robot tape system storage class, which in turn is followed by a large data storage capacity, but relatively slow data transfer tape robot system storage class. 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. Typically, files are first recorded at the top level of the storage hierarchy, then migrate to lower levels over time, following the rules laid down in the migration policy associated with the storage hierarchy. The purge policy determines when a file migrated to a lower level is purged from an upper level. When the file is read, it is usually first staged back to a higher level following the rules of the migration policy.

1.2.4. File family

A file family is an attribute of an HPSS file that is used to group files on tape virtual volumes. When a file is migrated from disk to tape, it is migrated to a tape virtual volume assigned to the family associated with the file. If no tape is associated with the family, the file is migrated to the next available tape not associated with a family, then that tape is assigned to the family. The family affiliation is preserved when tapes are repacked. Configuring file families is an HPSS 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.

1.3.1. Possible reasons for using FTP

- Utilizes standard FTP interface - Users and applications familiar with FTP can access HPSS with the standard command set.
- Supports standard FTP client on most platforms - FTP commands may be issued from any vendor nodes with an FTP interface.

Some GUI-based FTP clients may *not* function properly.

No specialized code is required.

1.3.2. Possible reasons for using PFTP

- Provides faster file transfers than FTP - PFTP is a better performer than FTP (for large files) since it provides the capability to stripe data across multiple client data ports.
- Supports files greater than 2 GB - PFTP supports file sizes up to 2^{64} bytes.
- Supports partial file transfer - PFTP provides options on the **pget** and **pput** commands to perform partial file transfers. This is beneficial to users who want to extract pieces of large files.

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. Contact your HPSS administrator to set up your account.

Users calling the utilities described in this document may need to be properly logged in with valid credentials. If the site is using Kerberos authentication, this will require the user to issue the **kinit** function call to acquire credentials. If UNIX authentication is being used, some of the utilities will require the user to provide the user password in order to use the utility. The **hpssuser** utility with proper parameters should be used by the HPSS administrator to create the Kerberos or UNIX accounts.

1.5. User accounts

As mentioned in the previous section, the user utilities may require the user to have obtained valid Kerberos or UNIX credentials. The HPSS administrator should use the **hpssuser** utility to create the proper user accounts.

In order to create the credentials, the proper command must be issued. For example, if you are using Kerberos authentication, the following should be issued:

```
% kinit <principal name>
```

When this command is entered, the principal's identity is validated, and the network credentials are obtained. The user will be prompted for the password.

If UNIX authentication is being used, the **HPSS** utility will prompt for the user password. An example of this is running the **scrub** utility.

If using Kerberos authentication and the login context is no longer required, the following command may be used to destroy the login context and associated credentials:

```
% kdestroy
```

Another command which might be of interest to the user is:

- **klist** - list the primary principal and tickets held in the Kerberos credentials cache.

Chapter 2. File Transfer Protocol (FTP)

This chapter describes the HPSS FTP interface. FTP is supported from any FTP client platform.

HPSS supports the FTP command set for transferring files to and from HPSS through the use of the following syntax:

```
ftp <host> [<port_number>]
```

where,

host is the name of the node where the HPSS PFTP 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 `451 Local resource failure: audit info.` 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. FTP site (and quote) commands

HPSS also supports the **site** commands listed below (for example, `site setcos 300` or `site setcos 300`). The **site** commands supported only by the HPSS PFTP clients are listed and described in Chapter 3, *Parallel File Transfer Protocol (PFTP)*.

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

- **abortreq**
- **allo64**
- **chgid**
- **chgrp**
- **chmod**
- **chown** (valid only for *root* account)
- **chuid** (valid only formatting *root* account)
- **getacct**
- **getfam**
- **feat**

- **genreqid**
- **idle**
- **mlst**
- **newer**
- **opts**
- **reportreqid**
- **rstagebatch**
- **setacct**
- **setcos**
- **seteomoncontrol**
- **setfam**
- **sethash**
- **stage**
- **stagebatch**
- **symlink**
- **udaset**
- **udaget**
- **wait**
- **umask**

A detailed explanation of each command follows.

2.1.1. Aborting a transfer by request ID - abortreq

This command is used to abort a file transfer by passing the transfer's short-form request ID, or a partial request ID.

```
site abortreq <request ID>
```

where,

request ID is the short-form (no dashes) string version of the request ID corresponding to a parallel file transfer. If a partial request ID is used, it must match only one request ID or the abort will fail.

Due to the underlying API calls involved, this functionality is only supported when cancelling parallel file transfers (**pput**/**pget**/PSTO/PRTR).

Example: The following will cancel the transfer associated with request ID f20885034774514db8d9a9ab489bb0f9

```
ftp> site abortreq f20885034774514db8d9a9ab489bb0f9
```

2.1.2. Allocating space for files - site allo64

This command is used to specify the size of a file for space allocation.

```
site 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 of a decimal number followed by a magnitude representation string. No spaces are allowed between the decimal number and the magnitude representation string. Accepted magnitude representation strings are:

KB (kilobyte = 1024),

MB (megabyte = 1,048,576),

GB (gigabyte = 1,073,741,824),

TB (terabyte = 1,099,511,627,776),

PB (petabyte = 1,125,899,906,842,624).

The magnitude representation string is case-independent. The decimal component may contain up to two decimal points of precision.

"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 **site allo** *size* command. The **site 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 GB.

```
ftp> site allo64 8GB
```

2.1.3. Changing a file’s group by ID - chgid

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

```
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".

```
ftp> site chgid 210 myfile
```

2.1.4. Changing a file’s group by name - chgrp

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

```
site chgrp <group> <file>
```

where,

group is the new group name 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 of `myfile` to group "mygroup".

```
ftp> site chgrp mygroup myfile
```

2.1.5. Changing a file’s permissions - chmod

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

```
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 bitwise 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

The following mode values are *only* supported by issuing the command using the symbolic formats:

`4000` set user ID on execution (u+s)

`2000` set group ID on execution (g+s)

`1000` sticky bit (o+t)

The *setuid* bit and sticky bits have no meaning in HPSS, but may be set and unset. The *setgid* bit is ON by default (although it is *not* printed by default) and sets "inheritance" when on.

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".

```
ftp> site chmod 0660 myfile
```

Example: The following may be entered to change the mode of *myfile* to "read", "write by owner", and "group", and also to set the gidbit. All other mode bits are left unchanged.

```
ftp> site chmod u+rw,g+rws myfile
```

2.1.6. Changing a file’s owner by name - chown

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

```
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".

```
ftp> site chown jones /home/smith/myfile
```

2.1.7. Changing a file’s owner by ID - chuid

This command is used to change the UID of a file and has the following format:

```
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".

```
ftp> site chuid 201 /home/smith/myfile
```

2.1.8. Generate a request ID that will be used on a future transfer - genreqid

This command is used to generate and return to the user a request ID that will be used on the next parallel (**pput**/**pget**/PSTO/PRTR) transfer by that user:

```
site genreqid
```

Each time a request ID is generated with this command, any previously generated request ID will be replaced by the new request ID. This means previously generated request IDs will not be used on the next parallel transfer and therefore will not be valid for use with the **abortreq** site command.

Example:

```
ftp> site genreqid
200 Transfer request identifier set to:  f20885034774514db8d9a9ab489bb0f9
```

2.1.9. Getting the account ID of a file or session - getacct

This command is used to get the account ID associated with a file or directory, or the current FTP session:

```
site getacct [filename]
```

where,

filename is the name of the file or directory for which the account ID is desired.

Example: The following will display the account ID of `myfile`

```
ftp> site getacct myfile
200 myfile account set to 12345
```

If no *filename* is specified, the command returns the current account ID setting for this FTP session. Example:

```
ftp> site getacct
200-Account set to 12345
200 Default account is 9876
```

2.1.10. Getting the file family of a file - getfam

This command is used to get the file family of a file:

```
site getfam [filename]
```

where,

filename is the name of the file for which the file family is desired.

Example: The following will display the file family of `myfile`

```
ftp> site getfam myfile
200 myfile file family set to 8
```

If no *filename* is specified, the command returns the current file family setting. Example:

```
ftp> site getfam
200 Current file family is 8.
```

2.1.11. Listing supported server features - feat

This command is list supported server features and has the following format:

```
quote feat
```

Note that due to a command naming conflict, the PFTP client must use *quote feat* rather than *feat* to get FTP server feature output.

Example:

```
ftp> quote feat
211-Extensions Supported:
 MDTM
 SIZE
 REST STREAM
 UTF8
 Type*;Size*;Create*;Modify*;Perm*;Family*;x.hpssvol;x.hpsshash;
 STAGEBATCH
 HPSS
211 End of extensions
```

2.1.12. Listing or setting idle time

This command is used to determine or change the default idle time (in seconds):

```
site idle time
```

where,

time is a number in seconds less than the maximum idle time.

Example: The following may be entered to set the idle time to 1000 seconds.

```
ftp> site idle 1000
200 Maximum IDLE time set to 1000 seconds
```

Without a size specifier, the current idle time is returned:

```
ftp> site idle
200 Current IDLE time limit is 1000 seconds; max 7200
```

2.1.13. Provide an easily parseable set of file facts - mlst

This command retrieves a semi-colon delimited list of information (also called facts) about a single provided file name. MLST by default returns a basic set of information about the file; however, there may be additional attributes supported by the server. These additional attributes likely will incur additional cost to retrieve, and can be identified by sending a "site feat" command. This will list the attributes which are available to be returned by MLST.

The command syntax is as follows:

```
ftp> quote mlst file-path
```

Example: The following will cause the MLST command to return only file size information:

```
ftp> quote mlst testfile
250-Entry fact information follows
 Type=file;Size=1024;Family=1;Modify=20190514103437;Create=20190514103435;Perm=awrdf; tes

250 MLST command successful.
```

2.1.13.1. MLST options

MLST output can be modified by sending an OPTS MLST command. The file name is always part of the output.

For example, to retrieve only the file size and HPSS volume information, send:

```
ftp> quote opts mlst size;x.hpssvol
200 MLST OPTS Size;x.hpssvol;
```

2.1.13.2. MLST HPSS-specific facts

Here is an explanation of the format of HPSS-specific facts. Other facts are described in the RFC documentation (https://tools.ietf.org/html/rfc3659).

x.hpssvol
: This fact describes the location of the file on the top-level tape tier. For example, if the hierarchy contains a disk at the top level followed by two tiers of tape, the first tape tier would be referenced. If dual copy is employed, the first copy is referenced. The fact describes the tape volume, the tape section, and the tape section offset of the file. These three pieces of information are delimited by a colon, for example:

```
ftp> quote mlst testfile
250-Entry fact information follows
 x.hpssvol=E0100800:1:0; testfile

250 MLST command successful.
```

x.hpsshash
: This fact describes the file hash information associated with the file in HPSS. This is not related to the file hash user attributes which may have been employed by a site. The format is the hash type,

followed by the creator string, followed by a hex representation of the checksum digest, followed by the file hash flags. The file hash flag values are described in detail in the HPSS Programmer's Reference.

For example:

```
ftp> quote mlst hash
250-Entry fact information follows
 x.hpsshash=md5:yourapp:9a8b41ad02b5d5d7c8fc8f38bf8af47d:USER|VALID; hash

250 MLST command successful.
```

Description of valid flags

1. USER - A user-supplied checksum (as opposed to one generated by HPSS automatically)
2. GENERATED - The file checksum was generated by HPSS (as opposed to provided by the user)
3. VALID - Checksum is not considered invalid - this does not mean it has been verified, just that it is not in an invalid state
4. VERIFIED - The user checksum has been verified by HPSS.
5. SKIP - The file has been flagged to skip verification during migration

2.1.14. File listings for files newer than a specified date

The **newer** command is used to determine which files in a directory are newer than a specified date. It provides a recursive short listing of files newer than the specified date.

```
site newer date path
```

where,

date refers to the date in the format YYYYMMDDHHMMSS

path refers to the pathname.

Example: The following may be entered to determine the files newer than March 03, 2005 13:51:22 in the path specified by *mypath*.

```
ftp> site newer 20050301135122 mypath
mypath/scrub_tests/file87
mypath/scrub_tests/file88
mypath/scrub_tests/file89
```

2.1.15. Modify server command behavior - opts

This command modifies the behavior of subsequent commands to the server.

List of commands which support OPTS:

1. MLST

For a reference of options for each supported command, refer to that command's documentation.

The command syntax is as follows:

```
quote opts command command-options
```

Example: The following will cause the MLST command to return only file size information:

```
ftp> quote opts mlst size
200 MLST OPTS Size;
```

2.1.16. Toggle display of request ID during transfers - reportreqid

This command acts as a toggle (default is off) that, when on, will report the request ID used for a parallel file transfer during that transfer.

```
site reportreqid
```

Example:

```
ftp> site reportreqid
200 Report request ID on.
ftp> pget hpsstest
200 Command Complete (110100480, "hpsstest", 0, 1, 4194304).
remote: hpsstest local: hpsstest
200 Command Complete.
150 Transfer starting. (Request ID f20885034774514db8d9a9ab489bb0f9)
...
```

2.1.17. Staging a batch of files recursively - rstagebatch

This command is used to initiate a stage of a batch of migrated files (for example, from tape to disk). The user can initiate the **rstagebatch** and then return at a later time to initiate the file transfer using the FTP **mget** or PFTP **mpget** commands.

This operation is recursive and will recursively stage all files in directories that match the glob pattern. For a nonrecursive batch stage, use **stagebatch**.

```
site stagebatch <glob>
```

where,

files is a glob to match the files to be staged.

Example: The following may be entered to stage files that match `/home/smith/f*`.

```
ftp> site stage /home/smith/f*
```

2.1.18. Setting the account ID of a file - setacct

This command is used to set the account ID associated with a file or directory

```
site setacct <filename> <account_id>
```

where,

filename is the name of the file or directory for which the user wishes to change the account ID.

account_id is the account ID the user wishes to assign to the file.

Example: The following will set the account ID of `myfile` to `12345`

```
ftp> site setacct myfile 12345
200 myfile account set to 12345
```

2.1.19. Specifying a file's Class of Service - setcos

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

```
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 media stripe width for a file. See your HPSS administrator for the Class of Service identifiers defined for your site. If a Class of Service is not specified, a default is used.

Parallel and striped transfers are *not* supported by the non-parallel FTP clients.

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".

```
ftp> site setcos 4
```

2.1.20. Specify that write operations should stop on EOM

This command is used to change the behavior of a parallel put operation. When seteomcontrol is enabled, if the write operation ends in EOM then the write will fail rather than continuing on a new device.

```
site seteomoncontrol <enable_flag>
```

where,

enable_flag is either 0 (for off) or non-zero (for on). The default value is 0. If no flag is passed, the server will reply with the current value.

When seteomoncontrol is enabled, it causes the HPSS core server to return control back to the client after data being written via a parallel transfer hits EOM. This value has no meaning for files being read.

In the example below, the following commands might be entered to cause an EOM to be returned to the client.

```
ftp> site eomoncontrol 1
200 EOMOnControl flag set to 1.
```

2.1.21. Specifying a file's file family - setfam

This command is used to specify a file family and has the following format:

```
site setfam [fam_id]
```

where,

fam_id is the file family identifier (used when creating a new HPSS file during a **put** operation).

Example: the following command might be entered to **put** a large file to HPSS with a file family of "8".

```
ftp> site setfam 8
200 Family set to 8.
```

If no *fam_id* is specified, the command returns the current file family setting. Example:

```
ftp> site setfam
200 Current file family is 8.
```

In order to clear the current file family, call this command with *fam_id* set to "0". Example:

```
ftp> site setfam 0
200 File family is now cleared
```

2.1.22. Specifying a file's file checksum - sethash

This command is used to specify a file checksum and has the following format:

```
site sethash <filename> <creator-name> <hash-type> <hash-string>
```

where,

filename is the file to modify.

creator-name is a user-specified creator string (for example, the application that generated the checksum).

hash-type is the type of checksum (for example, md5, sha512, and so on)

hash-string is a hex string checksum of the type, using the checksum type specified (for example, 9a8b41ad02b5d5d7c8fc8f38bf8af47d)

Example:

```
ftp> sethash testperm ftp md5 29a87a6da6ec8621156d11e611311c6d
250 SETHASH command successful.
```

2.1.23. Staging a file - stage

This command is used to initiate a stage of a migrated file (for example, 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.

```
site stage <file>
```

where,

file is the name of the file.

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

```
ftp> site stage /home/smith/myfile
```

2.1.24. Staging a batch of files - stagebatch

This command is used to initiate a stage of a batch of migrated files (for example, from tape to disk). The user can initiate the **stagebatch** and then return at a later time to initiate the file transfer using the FTP **mget** or PFTP **mpget** commands.

This operation is not recursive and will only stage the files in the top level. For a recursive batch stage, use **rstagebatch**.

```
site stagebatch <files>
```

where,

files is a glob to match the files to be staged.

Example: The following may be entered to stage files that match `/home/smith/f*`.

```
ftp> site stage /home/smith/f*
```

2.1.25. Creating a symbolic link - symlink

This command is used to create a symbolic link.

```
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 named sys_passwd in the local directory pointing to `/etc/passwd`.

```
ftp> site symlink /etc/passwd sys_passwd
```

A **dir** command will show `sys_passwd -> /etc/passwd`.

2.1.26. Retrieve a UDA attribute

This command is used to get a UDA attribute.

```
site udaget <path> <key> <format>
```

where,

path refers to the file

key refers to the UDA attribute

format refers to either *XML* or *value*.

Example: The following may be entered to retrieve a UDA attribute from `/hpss_test/file`.

```
ftp> site udaget /hpss_test/file /hpss/key value
200 PORT command successful.
150 Opening ASCII mode data connection for /hpss_test/file.
/hpss/key=   5
200 UDA attribute retrieved.
18 bytes received in 0.0124 seconds (0.001 MBytes/sec)
```

Example: The following may be entered to retrieve a UDA attribute in XML format from `/hpss_test/file`.

```
ftp> site udaget /hpss_test/file /hpss/key xml
200 PORT command successful.
150 Opening ASCII mode data connection for /hpss_test/file.
/hpss/key=   <key>5</key>
200 UDA attribute retrieved.
35 bytes received in 0.0089 seconds (0.004 MBytes/sec)
```

2.1.27. Set a UDA attribute

This command will set a UDA attribute.

```
site udaset <path> <key> <value>
```

where,

path refers to the file

key refers to the UDA attribute

value refers to the value of the UDA attribute. UDA attributes must start with */hpss*.

Example: The following may be entered to set a UDA attribute on `/hpss_test/file`.

```
ftp> site udaset /hpss_test/file /hpss/key 5
200 PORT command successful.
```

```
150 Opening ASCII mode data connection for /hpss_test/file.
/hpss_test/file   /hpss/key=   5
200 UDA attribute successfully set.
43 bytes received in 6.1855 seconds (0.000 MBytes/sec)
```

2.1.28. Setting the desire wait options (for migrated files) - wait

This command is used to notify the HPSS PFTP Daemon.

```
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** or **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** or **pget** command.

n (where n is an integer) - wait the specified period (in seconds) for the file requested by a **get** or **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.

```
ftp> site wait -1
```

The following table describes the behavior the customer should expect from FTP when issuing the **stage** and **wait** commands. Only Classes of Service utilizing the "Stage on Open Background" option will exhibit predictable results. Refer to the *Class of Service* section in the *HPSS Installation Guide* for more information.

Table 2.1. Behavior of stage and wait commands

Wait Time	File Condition	Command	Behavior/Message
No Wait	Archived	site stage xyz	"File xyz is being retrieved from archive."
No Wait	Not Archived	site stage xyz	"File xyz is currently ready for other processing."
Wait #	Archived	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	site stage xyz	"File xyz is currently ready for other processing."

Wait Time	File Condition	Command	Behavior/Message
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.

2.1.29. Changing the default umask

The **umask** command is used to change the default umask.

```
site umask octal-mask
```

where,

octal-mask refers to the octal mask to be applied.

Example: The following may be entered to change the default umask to "2".

```
ftp> site umask 0002
```

When issued without an octal value, the active umask is displayed.

2.2. List directory extensions

FTP supports the **ls** command to list the contents of a directory. The following standard options are supported: **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:

```
ftp> 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. In order to use PFTP, the PFTP client code must be compiled and supported on the client platform. Contact HPSS support for more guidance. PFTP supports the FTP command set plus some additional commands. Refer to *Section 3.3, "Additional PFTP commands"* for more information. 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###] [-Astring][-BStringSize] [-C###]
[-R#-#][-SsizeString][-T][-3][Host [Port]]
```

where,

Table 3.1. PFTP client command-line options

Option	Description
-b	Sets the PDATA protocol Blocksize. StringSize is the size specification in the format: Digit(s)Magnitude; for example, "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	Specifies use of 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.

Option	Description
-p	Specifies use of 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.
-A	Specify the user authentication mechanism. Values: GSS, USER_PASS
-B	Sets the Parallel Blocksize for parallel transfers. StringSize is the size specification in the format: Digit(s)Magnitude; for example, "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.
-P	Specifies use of the PDATA_PUSH protocol. This will be overridden if another protocol is specified in the `.netrc` file or if an explicit specification of another protocol is made by the user.
-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".
-S	Sets the maximum open/close socket size for HPSS parallel transfers. An artificial maximum of 250 GB (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 1.5 GB. StringSize is the size specification in the format: Digit(s)Magnitude; for example, "200GB".
-T	Terminate session on error. Useful for batch processing.
Host	The node where the HPSS PFTP daemon process resides.
Port	The port number for the HPSS PFTP daemon, as set in `/etc/services`.

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

The HPSS 7.5 **pftp_client** provides both username/password authentication and the GSS facilities previously provided by the **krb5_gss_pftp_client** binary in previous versions of HPSS. Contact HPSS support for details. These clients utilize either standard username/password authentication or the MIT Kerberos GSS facilities for authentication and reply processing. The GSS-based clients provide credential authentication facilities (password-less authentication) between the client and the HPSS PFTP daemon using Kerberos credentials for authentication.

The **pftp_client** does *not* obtain the end user's initial Kerberos credentials. The end user should obtain these credentials prior to initiating a **pftp_client** session by doing the

appropriate Kerberos **kinit** command. MIT Kerberos is available from MIT and will *not* be supplied by the HPSS project for non-HPSS platforms.

*Incompatibilities may exist between the HPSS (GSS) PFTP client and daemon and the Kerberos-based FTP features provided by IBM with AIX 4.x and AIX 5.x. This has been reported in the past and was outside the jurisdiction of the HPSS project. The HPSS PFTP clients and daemon **are** compatible with the MIT GSS-based FTP processes.*

As a courtesy to HPSS customers, the PFTP client code is available for compilation at customer sites upon request on non-HPSS platforms, subject to any legal or licensing restrictions. This explicitly denies any support requirement on IBM or the HPSS development and support personnel for any modifications made by the customer. Contact HPSS support for details.

The HPSS PFTP client has been successfully compiled on:

- Cray UNICOS [discontinued]
- Hewlett-Packard HP-UX [discontinued]
- Silicon Graphics IRIX [discontinued]
- Sun Solaris (Sparc and Intel)
- Intel Teraflop [discontinued]
- NEC [discontinued]
- IA64 Linux
- Linux Intel (RHEL 4.0 and 5.0)
- SUSE 9.x for AMD64
- Compaq Alpha [discontinued]
- IBM AIX 5.3 and 6.1.

All current ports may be compiled in either 32-bit or 64-bit mode.

Ports to other hardware or 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 or software ports into the distribution source. No site-specific features (local mods) added to the PFTP client by customer sites will be incorporated into the PFTP client without the appropriate modification to the HPSS license agreement.

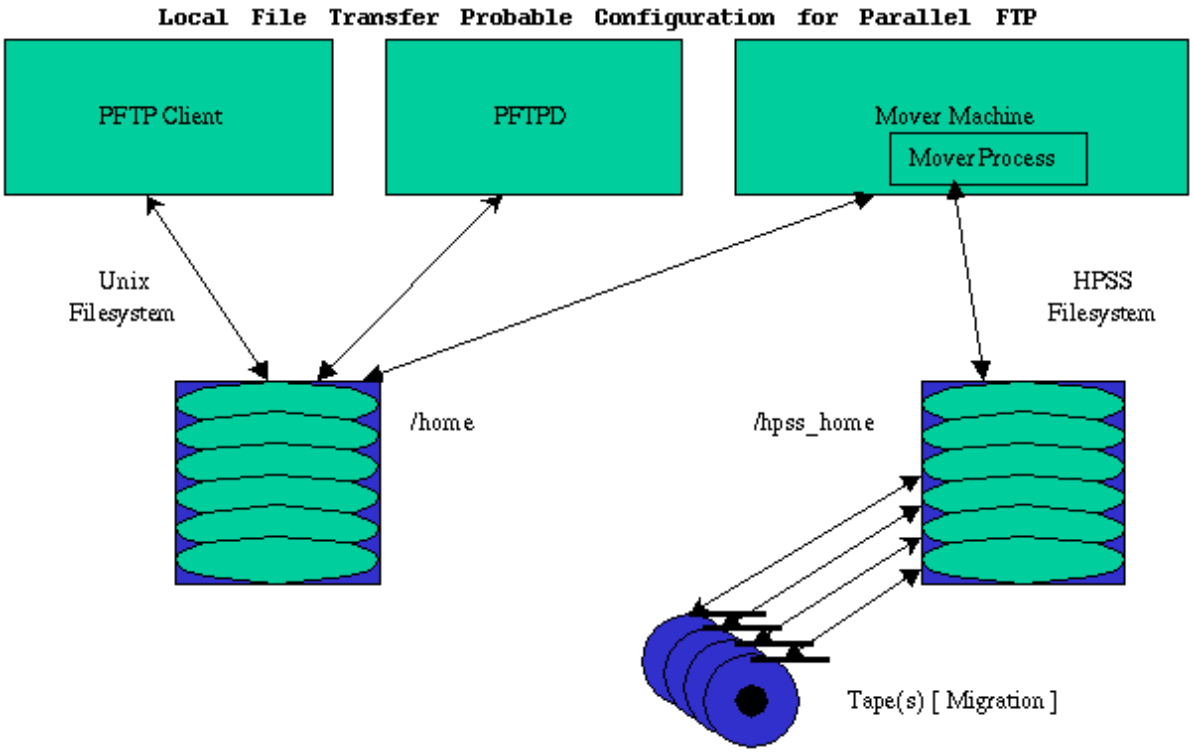
If the message `Local resource failure: audit info.` is received, contact your HPSS administrator. This message generally implies that either HPSS is not correctly configured, or some HPSS components may not be executing.

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 locally 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 same client node using one or more network interfaces (NICs).

Figure 3.1. Local File Transfer Probable Configuration for PFTP



The multinode option supports spawning the client processes across multiple machines/nodes and 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.

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 behavior and can result in data integrity problems.

The client nodes which participate in a multinode transfer are selected from the `HPSS.conf` configuration file (see discussion below) 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 which is maintained by the PFTP client.

3.1.1. Parallel FTP client/server configuration

The PFTP client can read configuration information from the `HPSS.conf` file to provide optimal performance characteristics. This file is configurable by HPSS administrators to provide performance enhancement. Performance enhancements require considerable expertise. Contact HPSS support if you need these features. The PFTP server also reads this file and the file should be present in the directory `/var/hpss/etc` along with other HPSS configuration files. It should be marked as readable by everybody and writable only by *root*, for security reasons.

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. For details of the implementation of the `HPSS.conf` file, review the HPSS Install Guide or contact your HPSS administrator. This file is constantly being updated with additional features and enhancements.

3.1.1.2. Local file functions

The local file functions represent performance enhancements using the HPSS parallel protocols where both the HPSS file and the UNIX source or destination file are "globally available" to the Movers and the PFTP client processes, such as GPFS file systems. The local file path must be specified in the file: `/var/hpss/etc/hpss_mvr_localfilepath.conf`. The specified path must exist for each PFTP client/Mover machine.

Configuration Criteria:

Figure 3.2. Local file transfer optimal configuration for PFTP

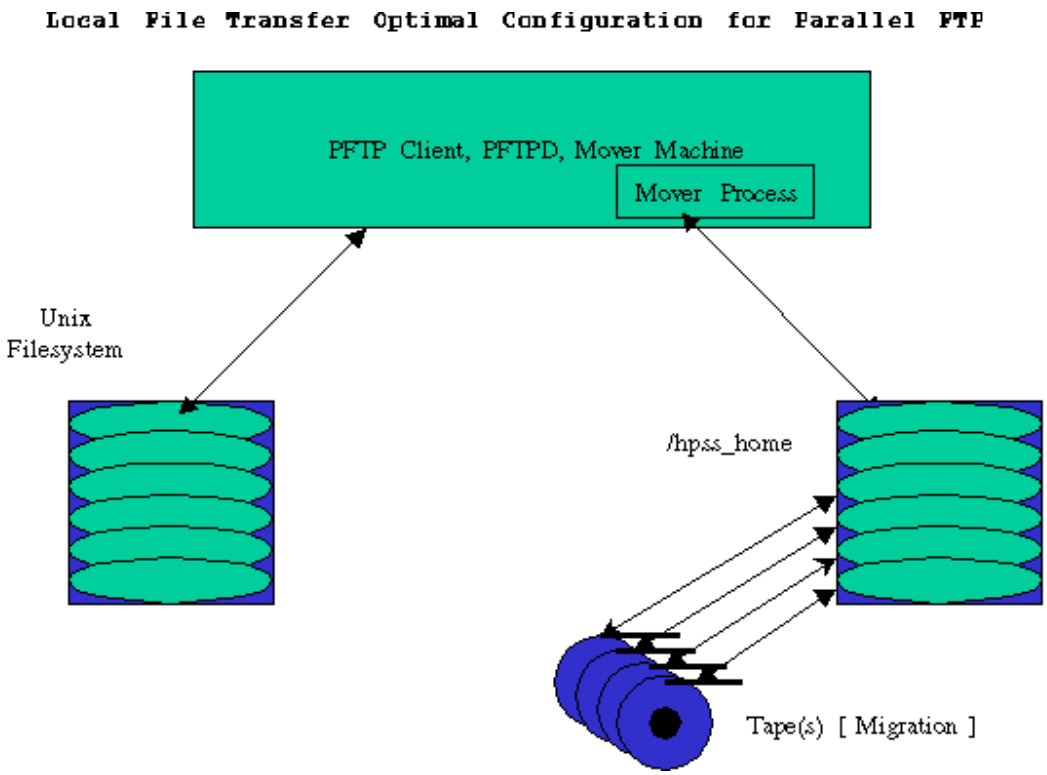
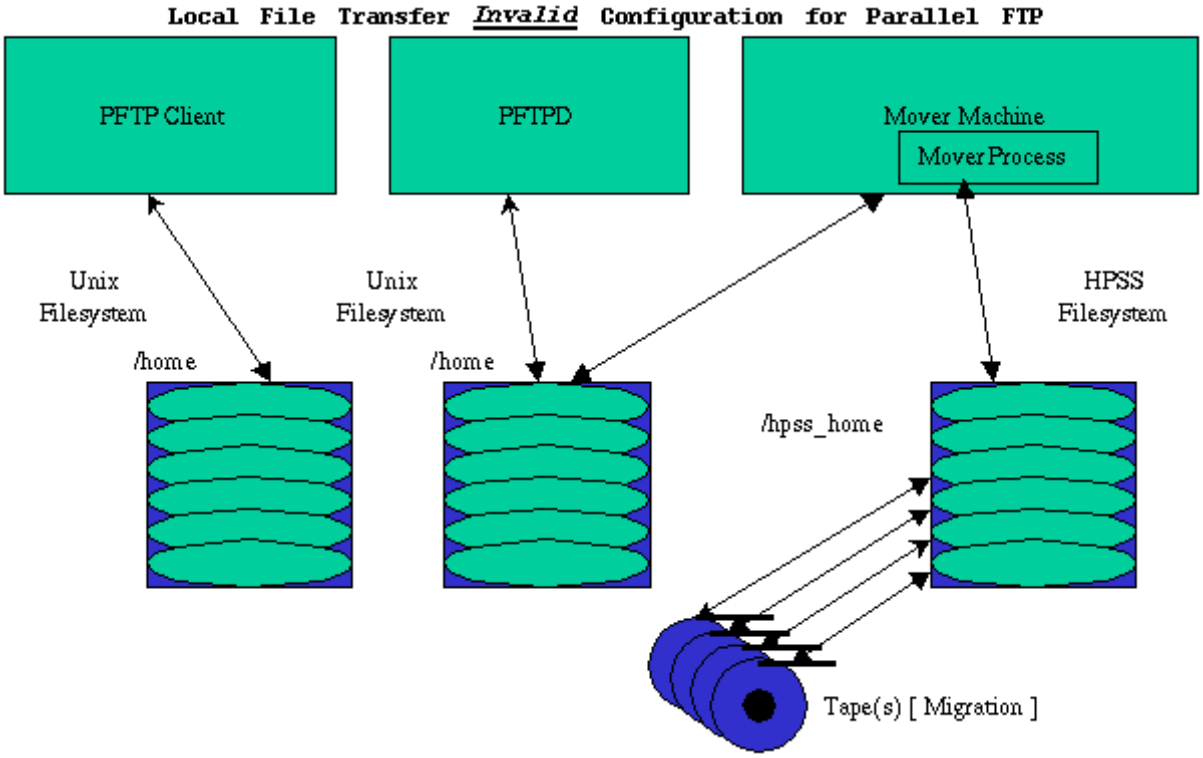


Figure 3.3. Local file transfer invalid configuration for PFTP



3.2. PFTP site (and quote) commands

Several additional site commands are available with PFTP.

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

- **acl**
- **bufsize**
- **gettun**
- **minfo**
- **rdrmt**
- **wtrmt**

These commands are described in the following subparagraphs.

3.2.1. Listing or setting HPSS ACLs

The **acl** command is used to list or set the ACLs for HPSS files and directories.

```
quote site acl command object
```

where,

command refers to an **acl** command from the following list:

- clear
- help
- remove
- replace
- show
- update

object refers to a file or directory.

Example: The following list the ACLs for a directory `st_data_mgmt_01`.

```
ftp> quote site acl show st_data_mgmt_01
200 PORT command successful.
150 Opening ASCII mode data connection for acl list.
 user_obj:rwxcid
 group_obj:rwx-id
 other_obj:r-x---
226 Transfer complete.
56 bytes received in 0.0006 seconds (0.093 Mbytes/sec)
```

3.2.2. Determining or setting buffer sizes [GridFTP]

The **bufsize** command is used to determine or set the buffer size to be used for parallel transfers.

```
quote site bufsize size
```

where,

size refers to the desired buffer size.

Example: The following may be entered to set the buffer size to 1 MB.

```
ftp> quote site bufsize 1048576
200 Current TCP Buffer size is 1048576
```

Without a size specifier, the current buffer size is returned ("-1" means use the default).

3.2.3. Reading configuration options for the PFTP server

The **quote gettun** command is used to determine characteristics that the PFTP server uses to transfer files to the client. This is not implemented at this time.

```
quote gettun
```

Example: The following lists the default characteristics of the PFTP Server.

```
ftp> quote gettun
```

3.2.4. File listings for files newer than a specified date

The **minfo** command is used to determine which files in a directory are newer than a specified date. It provides a recursive long listing of files and directories in the specified path.

```
quote site minfo date path
```

where,

date refers to the date in the format YYYYMMDDHHMMSS

path refers to the pathname.

Example: The following may be entered to determine the files newer than March 03, 2005 13:51:22 in the path specified by `mypath`.

```
ftp> quote site minfo 20050301135122 mypath
F Mon Mar 21 16:43:16 2005 mypath/scrub_tests/file87
F Mon Mar 21 16:43:25 2005 mypath/scrub_tests/file88
F Mon Mar 21 16:43:33 2005 mypath/scrub_tests/file89
```

3.2.5. Perform media timing (eliminating the network transfer time)

The **quote rdrmt** command is used to time the reading of data from media by the Movers. Similarly, the **quote wtrmt** command is used to time the performance of writing data to the HPSS media. This is *not* implemented for HPSS PFTP Server.

```
quote rdrmt filename Media_read_size Quantity
quote wtrmt filename Media_write_size Quantity
```

These functions are for obtaining performance information only and are not for use by users.

3.3. Additional PFTP commands

All FTP extensions described in Chapter 2 are supported by PFTP. In addition, the following commands (with their abbreviations, if any, shown in parentheses) are supported by PFTP:

- **pappend** (**papp**)
- **pput**, **mpput** (**ppu**, **mpp**)
- **pget**, **mpget** (**pge**, **mpg**)
- **lfappend** (**lfa**)
- **lfput**, **mlfput** (**lfp**, **mlfp**)

- **lfget**, **mlfget** (**lfg**, **mlf**)
- **recursive** (**recu**)
- **setpwidth** (**setpw**)
- **setpblocksize** (**setpb**)
- **multimode** (**multi**)
- **autoparallel** (**autop**)
- **getprot** (**getp**)
- **gettuningparms** (**gettun**)
- **pdata**
- **pdatapush** (**pdatap**)
- **pmover** (**pmov**)
- **setsockbufsize** (**sets**)
- **setxferbufsize** (**setx**)

Pipes are supported by the **pput** *and* **pget** *commands if an appropriate configuration is performed.* **The pipes facility uses temporary files. Configuration of the pipe temporary file path in the PFTP client section of `HPSS.conf` is strongly advised.**

3.3.1. General login messages (examples)

Without valid Kerberos credentials:

```
my_prompt> pftp_client fire 4021
Parallel block size set to 1048576.
Connected to fire.clearlake.ibm.com.
220-#
#    HPSS 7.5 Parallel FTP Daemon on fire
#               coming from fire.clearlake.ibm.com
#

220 fire FTP server (HPSS 7.5 PFTPD V1.1.1 Mon Apr 4 06:36:09
CDT 2005) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
init_sec_context: (krb5) Miscellaneous failure: No credentials
cache found
init_sec_context: (krb5) Miscellaneous failure: No credentials
cache found
GSSAPI authentication failed
Name (fire:whrahe): hpss
331 GSSAPI user hpss is not authorized as hpss - Password
required.
Password: {abcdefgh}
230 User hpss logged in as hpss.
```

```
Remote system type is UNIX.
Using binary mode to transfer files.
**** NOTE: FTP Daemon supports feature discovery ****
**** NOTE: Server supports Parallel Features    ****
****       Auto-Parallel Substitution Enabled.  ****
**** NOTE: Protocol set to PDATA_AND_MOVER ****
**** NOTE: Daemon does NOT support Transfer Agent
Pwidth set to default(1).
Multinode is Disabled.
ftp>
```

With appropriate Kerberos credentials (as HPSS):

```
my_prompt> pftp_client fire 4021
Parallel block size set to 1048576.
Connected to fire.clearlake.ibm.com.
220-#
#    HPSS 7.5 Parallel FTP Daemon on fire
#              coming from fire.clearlake.ibm.com
#

220 fire FTP server (HPSS 7.5 PFTPD V1.1.1 Mon Apr 4 06:36:09
CDT 2005) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
Preauthenticated FTP to fire as whrahe:
232 GSSAPI user hpss@FIRE.CLEARLAKE.IBM.COM is authorized as hpss
230 User hpss@FIRE.CLEARLAKE.IBM.COM logged in as hpss.
Remote system type is UNIX.
Using binary mode to transfer files.
**** NOTE: FTP Daemon supports feature discovery ****
**** NOTE: Server supports Parallel Features    ****
****       Auto-Parallel Substitution Enabled.  ****
**** NOTE: Protocol set to PDATA_AND_MOVER ****
**** NOTE: Daemon does NOT support Transfer Agent
Pwidth set to default(1).
Multinode is Disabled.
ftp>
```

3.3.2. 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

local_file. Identification of the file to transfer on the local machine.

remote_file. 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.

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
   ```

3.3.3. 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 the 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.

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

Parameters

-l *local_offset*. Optional byte offset into the local file where the transfer is to begin.

-r *remote_offset*. Optional byte offset into the remote file where the data is to be placed.

-s *size*. Optional byte size of the amount of data to transfer.

local_file. Identification of the file to transfer on the local machine.

remote_file. 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 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.

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 1 MB from offset 1 MB 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).
   ```
   ftp> pput " | tar cf - ./test*" my_test.tar
   ```

The pipe facility uses temporary files. It is strongly advised to configure a location for temporary files used by the pipe facility.

3.3.4. Multiple 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.

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*
   ```

3.3.5. 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 the transfer. The *local_offset*, *remote_offset*, and *size* may be specified using a decimal and magnitude representation string. See *Section 2.1.2, "Allocating space for files - site allo64"* for use of this notation.

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

Parameters
-r *remote_offset*. Optional byte offset where the transfer is to begin in the remote file.

-l *local_offset*. Optional parameter where the data is transferred in the local file.

-s *size*. Optional number of bytes to transfer.

remote_file. Identification of the file to transfer from the remote (HPSS) host.

local_file. 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.

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 1 MB 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).
   ```
   ftp> pget my_test.tar " | tar xf -"
   ```

The pipe facility uses temporary files. It is strongly advised to configure a location for temporary files used by the pipe facility.

3.3.6. Multiple 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.

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*
   ```

3.3.7. Local file append - lfappend

Synopsis

lfappend *local_file* [*remote_file*]

Description

The **lfappend** is a performance-optimized PFTP 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 machines (*not* "globally available"), the transfer will fail because the Movers will *not* be able to locate the desired file. The difference between **lfappend** and **pappend** is that the Movers involved in the transfer will not use the network to move the data, thus providing improved performance. The Mover machines *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

local_file. Identification of the "globally available" file to transfer.

remote_file. 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.

Examples

1. Append "globally available" file `testfile` to the same file name in the user's HPSS home directory.
   ```
   ftp> lfappend testfile
   ```
   . . . information is 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 is returned by the PFTP daemon
   ```
   ftp>
   ```

3.3.8. 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 PFTP client command used to transfer a "globally available" file into HPSS using the "parallel" protocols. *If* the file is not available to the Mover machines (*not* "globally available"), the transfer will fail because the Movers will *not* be able to locate the desired file. The difference between **lfput** and **pput** is that the Movers involved in the transfer will not use the network to move the data, thus providing improved performance. The Mover machines *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 the transfer. The *local_offset*, *remote_offset*, and *size* may be specified using a decimal and magnitude representation string. See *Section 2.1.2, "Allocating space for files - site allo64"* for use of this notation. The normal **lfput** command functions just like the standard FTP **put** command and transfers an entire file.

Parameters

-l *local_offset*. Optional byte offset into the "globally available" file where the transfer is to begin.

-r *remote_offset*. Optional byte offset into the remote file where the data is to be placed.

-s *size*. Optional byte size of the amount of data to transfer.

local_file. Identification of the "globally available" file to transfer. (This *must* be available to Mover machines.)

remote_file. 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.

Examples

1. Transfer local file `testfile` in the current working directory of the client to the user's HPSS home directory.
   ```
   ftp> cd ~
   ```
   . . . information is returned by the PFTP daemon
   ```
   ftp> lfput testfile
   ```
   . . . information is 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 is returned by the PFTP daemon
   ```
   ftp>
   ```

3. Transfer 1 MB from offset 1 MB 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 is returned by the PFTP daemon
ftp>
```

3.3.9. 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 PFTP 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 machines (*not* "globally available"), the transfer will fail because the Movers will *not* be able to locate the desired file. The difference between **lfget** and **pget** is that the Movers involved in the transfer will not use the network to move the data, thus providing improved performance. The Mover machines *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 machines. 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 the transfer. The *local_offset*, *remote_offset*, and *size* may be specified using a decimal and magnitude representation string. See *Section 3.3.10, "Multiple local file store - mlfput"* for use of this notation.

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

Parameters

-r *remote_offset*. Optional byte offset where the transfer is to begin in the remote file.

-l *local_offset*. Optional parameter where the data is transferred in the local file.

-s *size*. Optional number of bytes to transfer.

remote_file. Identification of the file to transfer from the remote (HPSS) host.

local_file. 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.

Examples

1. Transfer HPSS file `/home/bob/prod1` to the user's local directory.
   ```
   ftp> lfget /home/bob/prod1
   ```
   . . . *information is 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 is 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 is returned by the PFTP daemon*
   ```
   ftp>
   ```

4. Transfer 1 MB 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 is returned by the PFTP daemon*
   ```
   ftp>
   ```

3.3.10. Multiple local file store - mlfput

Synopsis

mlfput *local_files*

Description

The **mlfput** is a performance-optimized PFTP client command used to transfer multiple "globally available" files into HPSS using the "parallel" protocols. *If* the one or more files are not available to the Mover machines (*not* "globally available"), the transfer will fail because the Movers will *not* be able to locate the desired files. The difference between **mlfput** and **mpput** is that the Movers involved in the transfer will not use the network to move the data, thus providing improved performance. The Mover machines *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.

Examples

1. Transfer all "globally available" files in the current directory to the user's HPSS home directory.
   `ftp> cd ~`
   . . . information is returned by the PFTP daemon
   `ftp> prompt` *<== this toggles file prompting*
   `...`
   `ftp> mlfput *`
   . . . information is returned by the PFTP daemon
   `ftp>`

2. Transfer all "globally available" files which begin with "test" in directory `/usr/bob` to the user's HPSS home directory.
   `ftp> cd ~`
   . . . information is returned by the PFTP daemon
   `ftp> mlfput /usr/bob/test*`
   . . . information is returned by the PFTP daemon
   `ftp>`

3.3.11. Multiple local file retrieval - mlfget

Synopsis
mlfget *remote_files*

Description
The **mlfget** is a performance-optimized PFTP 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 machines (*not* "globally available"), the transfer

will fail because the Movers will *not* be able to locate the desired file. The difference between **mlfget** and **mpget** is that the Movers involved in the transfer will not use the network to move the data, thus providing improved performance. The Mover machines *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.

Examples

1. Transfer all files in HPSS directory `/home/bob` to the "globally available" current working directory.
   ```
   ftp> mlfget /home/bob/*
   ```
   . . . information is 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 is returned by the PFTP daemon
   ```
   ftp>
   ```

3.3.12. Recursive commands - recursive

Synopsis

recursive *mget*|*mput*|*mdelete dir_name*

Description

The **recursive** command is used to recursively **mget**, **mput**, or **mdelete** a directory.

Parameters

mget|mput|mdelete. The recursive command to be performed. The commands perform similarly to regular **mget**, **mput**, and **mdelete**, except they perform recursively on all the files and subdirectories contained in the specified directory.

dir_name. The directory on which to perform the recursive action. For **mget** or **mdelete**, this will be the remote directory to get or delete. For **mput**, this is the local directory to put.

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.

Not Found failures: specified *dir_name* does not exist or is not a directory

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

- "-5": an I/O error occurred.

See also

RFC-3659.
RFC-0959.

Examples

1. Transfer local directory `my_dir1` and all of its contents to the user's HPSS home directory.
   ```
   ftp> recursive mput my_dir1
   ```
2. Transfer directory `my_dir1` and all of its contents from the user's HPSS home directory to the local file system.
   ```
   ftp> recursive mget my_dir1
   ```
3. Delete directory `my_dir1` and all of its contents from the user's HPSS home directory.
   ```
   ftp> recursive mdelete my_dir1
   ```

3.3.13. 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 to determine the stripe width.

Return strings

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

Error conditions

```
Bad width value [stripe width].
```

See also

RFC-0959.

Examples

1. Set the stripe width to "4".
   ```
   ftp> setpwidth 4
   ```

3.3.14. 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 or magnitude representation string. See *Section 2.1.2, "Allocating space for files - site allo64"* for use of this notation. The maximum block size is 16 MB.

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 one byte through 16 MB. The default block size is 256 KB.

A general guideline would be to set *block_size* to the virtual volume block size. Consult your HPSS administrator to determine 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.

Examples

Set the transfer block size to 8 MB.

1. `ftp> setpblocksize 8388608`

2. `ftp> setpblocksize 8MB`

3.3.15. Multinode enable or disable - multinode

Synopsis

multinode

Description

The **multinode** command is used to enable or disable the ability 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.

Examples

None.

3.3.16. Autoparallel enable or disable - autoparallel

Synopsis

autoparallel

Description

The **autoparallel** command is used to enable or disable the automatic mapping of non-parallel commands to parallel commands; that is, **get** maps to **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

1. ```
   ftp> autop
   Automatic Substitution of Parallel Commands Disabled
   ```

   or

   ```
   Daemon supports Parallel Features - Auto-Parallel Substitution Enabled
   ```

3.3.17. 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

Current Parallel Protocol is PDATA PUSH
```

Error conditions

```
?Invalid command ==> Older Client?
```

See also

None.

Examples

1. ```
   ftp> getprot
   Current Parallel Protocol is PDATA and MOVER to MOVER
   Current Parallel Protocol is PDATA ONLY
   Current Parallel Protocol is PDATA PUSH
   ```

3.3.18. Get tuning parameters - gettun

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)` (This may be observed at login time.)

HPSS.conf parsing errors.

See also

None.

Examples

1.
```
ftp> gettun
Effective Tuning parameters from saux22 to sair031
     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
```

The use of "Parallel Pipes" should be discouraged; however, you may observe either of the following two scenarios:
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 = /copylvol/.pftp_pipes26404
Pipe File Size = 1073741824

3.3.19. 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>
   ```

3.3.20. Set the PDATA_PUSH protocol - pdatapush

Synopsis
pdatapush

Description
Explicitly request the PDATA_PUSH protocol.

Parameters
None.

Return strings
`**** NOTE: Protocol set to PDATA_PUSH ****` (at logon time)
`215 Parallel protocol is PDATA_PUSH`

Error conditions
`?Invalid command ==> Older Client?`

See also
None.

Examples

1. Set protocol to PDATA_PUSH (failure).
```
ftp> pdatapush
Server does NOT support command ==> Older Server?
?Invalid command ==> Older Client?
ftp>
```

3.3.21. Set the PDATA_AND_MOVER protocol - pmover

Synopsis
pmover

Description
Explicitly specify parallel transfers to use the PDATA_AND_MOVER protocol regardless of what is specified in the `HPSS.conf` file.

Parameters
None.

Return strings
`215 Parallel protocol is PDATA_AND_MOVER`

Error conditions
`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>
```

3.3.22. Set the socket buffer size - setsock

Synopsis

setsock *SizeString*

Description

Set the desired socket buffer size. This is 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. For example, "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 (above system maximum).
   ```
   ftp> setsock 4mb
   PdataSockBufSize reset equal or below sb_max (1048576)
   ftp>
   ```

2. Set socket buffer size.
   ```
   ftp> setsock 512kb
   ftp>
   ```

3. Set socket buffer size (no argument).
   ```
   ftp> setsock
   Socket Buffer Size = 524288.
   ftp>
   ```

3.3.23. Set the transfer buffer size - setxfer

Synopsis

setxferbufsize *SizeString*

Description

Set the desired transfer buffer sizes. This is 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. For example, "4MB".

Return strings

```
PdataBufferSize = 4194304
```

Error conditions

```
?Invalid command ==> Old Client?
```

See also

None.

Examples

1. Set transfer buffer size (the maximum value is 32 MB).

```
ftp> setxfer 40MB
ftp>
```

2. Display effective transfer buffer size (no argument).

```
ftp> setxfer
PdataBufferSize = 33554432.
ftp>
```

Appendix A. Glossary of terms and acronyms

ACL	Access Control List
ACSLS	Automated Cartridge System Library Software (Oracle StorageTek)
ADIC	Advanced Digital Information Corporation
accounting	The process of tracking system usage per user, possibly for the purposes of charging for that usage. Also, a log record type used to log accounting information.
ACI	AML Client Interface
AIX	Advanced Interactive Executive. An operating system provided on many IBM machines.
alarm	A log record type used to report situations that require administrator investigation or intervention.
AML	Automated Media Library. A tape robot.
AMS	Archive Management Unit
ANSI	American National Standards Institute
API	Application Program Interface
archive	One or more interconnected storage systems of the same architecture.
ASLR	Address Space Layout Randomization
attribute	When referring to a managed object, an attribute is one discrete piece of information, or set of related information, within that object.
attribute change	When referring to a managed object, an attribute change is the modification of an object attribute. This event may result in a notification being sent to SSM, if SSM is currently registered for that attribute.
audit (security)	An operation that produces lists of HPSS log messages whose record type is SECURITY. A security audit is used to provide a trail of security-relevant activity in HPSS.
AV	Account Validation
bar code	An array of rectangular bars and spaces in a predetermined pattern which represent alphanumeric information in a machine-readable format (such as a UPC symbol).
BFS	HPSS Bitfile Service
bitfile	A file stored in HPSS, represented as a logical string of bits unrestricted in size or internal structure. HPSS imposes a size limitation in 8-bit bytes based upon the maximum size in bytes that can be represented by a 64-bit unsigned integer.
bitfile segment	An internal metadata structure, not normally visible, used by the Core Server to map contiguous pieces of a bitfile to underlying storage.
Bitfile Service	Portion of the HPSS Core Server that provides a logical abstraction of bitfiles to its clients.

BBTM	Blocks Between Tape Marks. The number of data blocks that are written to a tape virtual volume before a tape mark is required on the physical media.
CAP	Cartridge Access Port
cartridge	A physical media container, such as a tape reel or cassette, capable of being mounted on and dismounted from a drive. A fixed disk is technically considered to be a cartridge because it meets this definition and can be logically mounted and dismounted.
class	A type definition in Java. It defines a template on which objects with similar characteristics can be built, and includes variables and methods specific to the class.
Class of Service	A set of storage system characteristics used to group bitfiles with similar logical characteristics and performance requirements together. A Class of Service is supported by an underlying hierarchy of storage classes.
cluster	The unit of storage space allocation on HPSS disks. The smallest amount of disk space that can be allocated from a virtual volume is a cluster. The size of the cluster on any given disk volume is determined by the size of the smallest storage segment that will be allocated on the volume, and other factors.
configuration	The process of initializing or modifying various parameters affecting the behavior of an HPSS server or infrastructure service.
COS	Class of Service
control path	For the SCSI PVR, this is a connection to the library for sending commands. Control paths can be discovered using device_scan.
Core Server	An HPSS server which manages the name space and storage for an HPSS system. The Core Server manages the name space in which files are defined, the attributes of the files, and the storage media on which the files are stored. The Core Server is the central server of an HPSS system. Each storage subsystem uses exactly one Core Server.
CRC	Cyclic Redundancy Check
CS	Core Server
daemon	A UNIX program that runs continuously in the background.
DAS	Distributed AML Server
DB2	A relational database system, a product of IBM Corporation, used by HPSS to store and manage HPSS system metadata.
DCE	Distributed Computing Environment
debug	A log record type used to report internal events that can be helpful in troubleshooting the system.
delog	The process of extracting, formatting, and outputting HPSS central log records. This process is obsolete in 7.4 and later versions of HPSS. HPSS logs are now recorded as plain text.
deregistration	The process of disabling notification to SSM for a particular attribute change.
descriptive name	A human-readable name for an HPSS server.
device	A physical piece of hardware, usually associated with a drive, that is capable of reading or writing data.

directory	An HPSS object that can contain files, symbolic links, hard links, and other directories.
dismount	An operation in which a cartridge is either physically or logically removed from a device, rendering it unreadable and unwritable. In the case of tape cartridges, a dismount operation is a physical operation. In the case of a fixed disk unit, a dismount is a logical operation.
DNS	Domain Name Service
DOE	Department of Energy
DPF	Database Partitioning Feature
drive	A physical piece of hardware capable of reading or writing mounted cartridges. The terms device and drive are often used interchangeably.
EB	Exabyte (2^{60})
EOF	End of File
EOM	End of Media
ERA	Extended Registry Attribute
ESCON	Enterprise System Connection
event	A log record type used to report informational messages (for example, subsystem starting or subsystem terminating).
export	An operation in which a cartridge and its associated storage space are removed from the HPSS system Physical Volume Library. It may or may not include an eject, which is the removal of the cartridge from its Physical Volume Repository.
FC SAN	Fiber Channel Storage Area Network
FIFO	First in first out
file	An object than can be written to, read from, or both, with attributes including access permissions and type, as defined by POSIX (P1003.1-1990). HPSS supports only regular files.
file family	An attribute of an HPSS file that is used to group a set of files on a common set of tape virtual volumes.
fileset	A collection of related files that are organized into a single easily managed unit. A fileset is a disjoint directory tree that can be mounted in some other directory tree to make it accessible to users.
fileset ID	A 64-bit number that uniquely identifies a fileset.
fileset name	A name that uniquely identifies a fileset.
file system ID	A 32-bit number that uniquely identifies an aggregate.
FTP	File Transfer Protocol
FSF	Forward Space File
FSR	Forward Space Record
Gatekeeper	An HPSS server that provides two main services: the ability to schedule the use of HPSS resources referred to as the Gatekeeping Service, and the ability to validate user accounts referred to as the Account Validation Service.

Gatekeeping Service	A registered interface in the Gatekeeper that provides a site the mechanism to create local policy on how to throttle or deny create, open and stage requests and which of these request types to monitor.
Gatekeeping Site Interface	The APIs of the gatekeeping site policy code.
Gatekeeping Site Policy	The gatekeeping shared library code written by the site to monitor and throttle create, open, and/or stage requests.
GB	Gigabyte (2^{30})
GECOS	The comment field in a UNIX password entry that can contain general information about a user, such as office or phone number.
GID	Group Identifier
GK	Gatekeeper
GSS	Generic Security Service
GUI	Graphical User Interface
HA	High Availability
HACMP	High Availability Clustered Multi-Processing - A software package used to implement high availability systems.
HADR	DB2 High Availability Disaster Recovery
halt	A forced shutdown of an HPSS server.
HBA	Host Bus Adapter
HDM	Shorthand for HPSS/DMAP.
hierarchy	See *storage hierarchy*.
HPSS	High Performance Storage System
HPSS-only fileset	An HPSS fileset that is not linked to an external file system (such as XFS).
HTP	HPSS Test Plan
IBM	International Business Machines Corporation
ID	Identifier
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronics Engineers
import	An operation in which a cartridge and its associated storage space are made available to the HPSS system. An import requires that the cartridge has been physically introduced into a Physical Volume Repository (injected). Importing the cartridge makes it known to the Physical Volume Library.
I/O	Input/Output
IOD/IOR	I/O Descriptor/I/O Reply. Structures used to send control information about data movement requests in HPSS and about the success or failure of the requests.
IP	Internet Protocol
IRIX	SGI’s implementation of UNIX
JRE	Java Runtime Environment
junction	A mount point for an HPSS fileset.

KB	Kilobyte (2^{10})
KDC	Key Distribution Center
LAN	Local Area Network
LANL	Los Alamos National Laboratory
latency	For tape media, the average time in seconds between the start of a read or write request and the time when the drive actually begins reading or writing the tape.
LBP	Logical Block Protection
LDAP	Lightweight Directory Access Protocol
LFT	Local File Transfer
LLNL	Lawrence Livermore National Laboratory
LMU	Library Management Unit
Location Server	An HPSS server that is used to help clients locate the appropriate Core Server or other HPSS server to use for a particular request.
log record	A message generated by an HPSS application and handled and recorded by the HPSS logging subsystem.
log record type	A log record may be of type alarm, event, info, debug, request, security, trace, or accounting.
logging service	An HPSS infrastructure service consisting of the logging subsystem and one or more logging policies. A default logging policy can be specified, which will apply to all servers, or server-specific logging policies may be defined.
LS	Location Server
LSM	Library Storage Module
LTO	Linear Tape-Open. A half-inch open tape technology developed by IBM, HP, and Seagate.
LUN	Logical Unit Number
LVM	Logical Volume Manager
MAC	Mandatory Access Control
managed object	A programming data structure that represents an HPSS system resource. The resource can be monitored and controlled by operations on the managed object. Managed objects in HPSS are used to represent servers, drives, storage media, jobs, and other resources.
MB	Megabyte (2^{20})
MBS	Media Block Size
metadata	Control information about the data stored under HPSS, such as location, access times, permissions, and storage policies. Most HPSS metadata is stored in a DB2 relational database.
method	A Java function or subroutine.
migrate	To copy file data from a level in the file's hierarchy onto the next lower level in the hierarchy.
Migration/ Purge Server	An HPSS server responsible for supervising the placement of data in the storage hierarchies based upon site-defined migration and purge policies.

MM	Metadata Manager. A software library that provides a programming API to interface HPSS servers with the DB2 programming environment.
mount	An operation in which a cartridge is either physically or logically made readable/writable on a drive. In the case of tape cartridges, a mount operation is a physical operation. In the case of a fixed disk unit, a mount is a logical operation.
mount point	A place where a fileset is mounted in the XFS and HPSS name spaces.
Mover	An HPSS server that provides control of storage devices and data transfers within HPSS.
MPS	Migration/Purge Server
MVR	Mover
NASA	National Aeronautics and Space Administration
Name Service	The portion of the Core Server that provides a mapping between names and machine-oriented identifiers. In addition, the Name Service performs access verification and provides the Portable Operating System Interface (POSIX).
name space	The set of name-object pairs managed by the HPSS Core Server.
NERSC	National Energy Research Supercomputer Center
NIS	Network Information Service
NLS	National Language Support
notification	A notice from one server to another about a noteworthy occurrence. HPSS notifications include notices sent from other servers to SSM of changes in managed object attributes, changes in tape mount information, and log messages of type alarm or event.
NS	HPSS Name Service
NSL	National Storage Laboratory
object	See *managed object*.
ORNL	Oak Ridge National Laboratory
OS	Operating System
OS/2	The operating system (multi-tasking, single user) used on the AMU controller PC.
PB	Petabyte (2^{50})
PFTP	Parallel File Transfer Protocol
PFTPD	PFTP Daemon
physical volume	An HPSS object managed jointly by the Core Server and the Physical Volume Library that represents the portion of a virtual volume. A virtual volume may be composed of one or more physical volumes, but a physical volume may contain data from no more than one virtual volume.
Physical Volume Library	An HPSS server that manages mounts and dismounts of HPSS physical volumes.
Physical Volume Repository	An HPSS server that manages the robotic agent responsible for mounting and dismounting cartridges or interfaces with the human agent responsible for mounting and dismounting cartridges.
PIO	Parallel I/O

PIOFS	Parallel I/O File System
POSIX	Portable Operating System Interface (for computer environments).
purge	Deletion of file data from a level in the file's hierarchy after the data has been duplicated at lower levels in the hierarchy and is no longer needed at the deletion level.
purge lock	A lock applied to a bitfile which prohibits the bitfile from being purged.
PV	Physical Volume
PVL	Physical Volume Library
PVM	Physical Volume Manager
PVR	Physical Volume Repository
RAID	Redundant Array of Independent Disks
RAIT	Redundant Array of Independent Tapes
RAM	Random Access Memory
RAO	Recommended Access Order
reclaim	The act of making previously written but now empty tape virtual volumes available for reuse. Reclaimed tape virtual volumes are assigned a new Virtual Volume ID, but retain the rest of their previous characteristics. Reclaim is also the name of the utility program that performs this task.
registration	The process by which SSM requests notification of changes to specified attributes of a managed object.
reinitialization	An HPSS SSM administrative operation that directs an HPSS server to reread its latest configuration information, and to change its operating parameters to match that configuration, without going through a server shutdown and restart.
repack	The act of moving data from a virtual volume onto another virtual volume with the same characteristics with the intention of removing all data from the source virtual volume. Repack is also the name of the utility program that performs this task.
request	A log record type used to report some action being performed by an HPSS server on behalf of a client.
RISC	Reduced Instruction Set Computer/Cycles
RPC	Remote Procedure Call
RSF	Reverse Space File
RSR	Reverse Space Record
SCSI	Small Computer Systems Interface
security	A log record type used to report security-related events (for example, authorization failures).
SGI	Silicon Graphics
shelf tape	A cartridge which has been physically removed from a tape library but whose file metadata still resides in HPSS.
shutdown	An HPSS SSM administrative operation that causes a server to stop its execution gracefully.
sink	The set of destinations to which data is sent during a data transfer, such as disk devices, memory buffers, or network addresses.

SM	System Manager
SMC	SCSI Medium Changer
SME	Subject Matter Expert
SNL	Sandia National Laboratories
SOID	Storage Object ID. An internal HPSS storage object identifier that uniquely identifies a storage resource. The SOID contains a unique identifier for the object, and a unique identifier for the server that manages the object.
source	The set of origins from which data is received during a data transfer, such as disk devices, memory buffers, or network addresses.
SP	Scalable Processor
SS	HPSS Storage Service
SSD	Solid State Drive
SSH	Secure Shell
SSI	Storage Server Interface
SSM	Storage System Management
SSM session	The environment in which an SSM user interacts with the SSM System Manager to monitor and control HPSS. This environment may be the graphical user interface provided by the **hpssgui** program, or it may be the command-line user interface provided by the **hpssadm** program.
SSMSM	Storage System Management System Manager
stage	To copy file data from a level in the file's hierarchy onto the top level in the hierarchy.
start-up	An HPSS SSM administrative operation that causes a server to begin execution.
info	A log record type used to report file staging and other kinds of information.
STK	Storage Technology Corporation (Oracle StorageTek)
storage class	An HPSS object used to group storage media together to provide storage for HPSS data with specific characteristics. The characteristics are both physical and logical.
storage hierarchy	An ordered collection of storage classes. The hierarchy consists of a fixed number of storage levels numbered from level 1 to the number of levels in the hierarchy, with the maximum level being limited to 5 by HPSS. Each level is associated with a specific storage class. Migration and stage commands result in data being copied between different storage levels in the hierarchy. Each Class of Service has an associated hierarchy.
storage level	The relative position of a single storage class in a storage hierarchy. For example, if a storage class is at the top of a hierarchy, the storage level is 1.
storage map	An HPSS object managed by the Core Server to keep track of allocated storage space.
storage segment	An HPSS object managed by the Core Server to provide abstract storage for a bitfile or parts of a bitfile.
Storage Service	The portion of the Core Server which provides control over a hierarchy of virtual and physical storage resources.
storage subsystem	A portion of the HPSS name space that is managed by an independent Core Server and (optionally) Migration/Purge Server.

Storage System Management	An HPSS component that provides monitoring and control of HPSS via a windowed operator interface or command-line interface.
stripe length	The number of bytes that must be written to span all the physical storage media (physical volumes) that are grouped together to form the logical storage media (virtual volume). The stripe length equals the virtual volume block size multiplied by the number of physical volumes in the stripe group (that is, stripe width).
stripe width	The number of physical volumes grouped together to represent a virtual volume.
System Manager	The Storage System Management (SSM) server. It communicates with all other HPSS components requiring monitoring or control. It also communicates with the SSM graphical user interface (**hpssgui**) and command line interface (**hpssadm**).
TB	Terabyte (2^{40})
TCP/IP	Transmission Control Protocol/Internet Protocol
TDS	Tivoli Directory Server
TI-RPC	Transport-Independent-Remote Procedure Call
trace	A log record type used to record procedure entry/exit events during HPSS server software operation.
transaction	A programming construct that enables multiple data operations to possess the following properties: • All operations commit or abort/roll-back together such that they form a single unit of work. • All data modified as part of the same transaction are guaranteed to maintain a consistent state whether the transaction is aborted or committed. • Data modified from one transaction are isolated from other transactions until the transaction is either committed or aborted. • Once the transaction commits, all changes to data are guaranteed to be permanent.
TSA/MP	Tivoli System Automation for Multiplatforms
TSM	Tivoli Storage Manager
UDA	User-defined Attribute
UDP	User Datagram Protocol
UID	User Identifier
UPC	Universal Product Code
UUID	Universal Unique Identifier
VPN	Virtual Private Network
virtual volume	An HPSS object managed by the Core Server that is used to represent logical media. A virtual volume is made up of a group of physical storage media (a stripe group of physical volumes).
virtual volume block size	The size of the block of data bytes that is written to each physical volume of a striped virtual volume before switching to the next physical volume.

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

Appendix B. References

1. **File Transfer Protocol, RFC-0959**, October 1985.
2. **HPSS Error Manual**.
3. **HPSS Programmer's Reference**.
4. **HPSS Installation Guide**.
5. **HPSS Management Guide**.
6. **Installing, Managing, and Using the IBM AIX Parallel I/O File System**, Document Number H34- 6065-00.
7. **POSIX 1003.1-1990 Tar Standard**.

Appendix C. Developer acknowledgments

HPSS is a product of a government-industry collaboration. The project approach is based on the premise that no single company, government laboratory, or research organization has the ability to confront all of the system-level issues that must be resolved for significant advancement in high-performance storage system technology.

HPSS development was performed jointly by IBM Worldwide Government Industry, Lawrence Berkeley National Laboratory, Lawrence Livermore National Laboratory, Los Alamos National Laboratory, NASA Langley Research Center, Oak Ridge National Laboratory, and Sandia National Laboratories.

We would like to acknowledge Argonne National Laboratory, the National Center for Atmospheric Research, and Pacific Northwest Laboratory for their help with initial requirements reviews.

We also wish to acknowledge Cornell Information Technologies of Cornell University for providing assistance with naming service and transaction management evaluations and for joint developments of the Name Service.

In addition, we wish to acknowledge the many discussions, design ideas, implementation and operation experiences we have shared with colleagues at the National Storage Laboratory, the IEEE Mass Storage Systems and Technology Technical Committee, the IEEE Storage System Standards Working Group, and the storage community at large.

We also wish to acknowledge the Cornell Theory Center and the Maui High Performance Computer Center for providing a test bed for the initial HPSS release.

We also wish to acknowledge Gleicher Enterprises, LLC for the development of the HSI, HTAR, and Transfer Agent client applications.

Finally, we wish to acknowledge CEA-DAM (**Commissariat à l'Énergie Atomique - Centre d'Études de Bruyères-le-Châtel**) for providing assistance with development of NFS V3 protocol support.