HPSS
User's Guide

High Performance Storage System
Release 7.4

September 2012 *(Revision 1.0)*

Printed in the United States of America.

HPSS Release 7.4
September 2012 (Revision 1.0)

Table of Contents

List of Tables

Preface

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)

Note: 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.
- ***Chapter 4: Linux Virtual File System (VFS)*** - Defines the VFS 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 entitled: "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. HPSS Virtual File System (VFS) Interface

The HPSS VFS Interface is a client application that uses the VFS switch to present a standard POSIX file system interface. Users can access the HPSS disk cache as if it were a locally mounted file system. Support for getting and setting HPSS COS information is provided via file system specific ioctl() commands. See the *HPSS Programmers Guide* for more information about supported ioctl() extensions. See the *HPSS Installation Guide* and *HPSS Management Guide* for more information. Currently, this interface is only available for Linux clients.

1.1.4. User Utilities hacl and lshpss

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

- **hacl** - edit HPSS Access Control Lists.
- **lshpss** - list information concerning the HPSS configuration (Class of Service list, hierarchy list, storage class list, physical volumes, devices and drives, servers, Movers, and other metadata)

Refer to the man pages for details on use of these utilities.

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. Note: If the COS ID is explicitly set by using the **`site setcos`** command, that COS will be used regardless of file size.

A COS is implemented by a Storage Hierarchy of one to three 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 non-zero 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 a 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. *NOTE: 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 gigabytes - 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.

1.3.3. Possible Reasons for Using HPSS VFS Interface

- Provides a standard POSIX file system interface – This allows any client application, that utilizes a standard POSIX file system for storage, to store data in HPSS without having to rewrite the client application using the HPSS client library.
- Utilizes the Linux file system cache to keep recently accessed data in memory on the client machine.
- Provides a transparent user interface into HPSS.

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. To use FTP, use 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. Site (and Quote) Commands

HPSS also supports the site commands listed below (e.g., "**site setcos 300**" or "**quote site setcos 300**"). *NOTE: site commands supported only by the HPSS PFTP clients are listed and described in Chapter 3, Parallel File Transfer Protocol.*

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

- **allo64**
- **chgid**
- **chgrp**
- **chmod**
- **chown** (valid only for "root" account)
- **chuid** (valid only formatting "root" account)
- **idle**
- **newer**
- **setcos**
- **stage**
- **symlink**
- **wait**
- **umask**

A detailed explanation of each command follows.

2.1.1. Allocating space for files - quote allo64

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

quote allo64 <size>

where,

size is a string representing the size of the file. The size may be a decimal number less than 2^{64} or may be in the form 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 = 1048576),

GB (gigabyte = 1073741824),

TB (terabyte = 1099511627776),

PB (petabyte = 1125899906842624).

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

NOTE: 1005.03 will truncate to 1005 if no magnitude representation string is specified. Similar truncations will occur for excess precision specifications.

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

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

```
ftp> quote allo64 8GB
```

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

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

```
quote site chgid <gid> <file>
```

where,

gid is the new group ID of the file

file is the name of the file.

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

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

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

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

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

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

where,

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

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

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

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

2.1.4. Changing a File's Permissions - chmod

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

```
quote site chmod <mode> <file>
```

where,

mode is the new octal mode number of the file

file is the name of the file.

Mode is constructed from the OR of the following modes:

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

Note: The following mode values are 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)

NOTE: the setuid bit and sticky bits have no meaning in HPSS; but may be set/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> quote 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 set the gidbit. All other mode bits are left unchanged.

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

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

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

```
quote site chown <owner> <file>
```

where,

owner is the new owner of the file

file is the name of the file.

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

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

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

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

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

```
quote site chuid <uid> <file>
```

where,

uid is the new uid of the owner of the file

file is the name of the file.

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

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

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

2.1.7. Listing/Setting Idle Time

Used to determine / change the Default Idle Time (in seconds)

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

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

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

2.1.8. File Listings for files Newer than a Specified Date

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

```
quote site newer date path
```

where,

date refers to the date in the format YYYYMMDDHHMMSS and

path refers to the a 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 newer 20050301135122 mypath
mypath/scrub_tests/file87
mypath/scrub_tests/file88
mypath/scrub_tests/file89
```

2.1.9. Specifying a File's Class of Service - setcos

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

```
quote site setcos <cos_id>
```

where,

cos_id is the Class of Service identifier (used when creating a new HPSS file during a put operation.)

Class of Service is used as a means for specifying the amount of parallelism or 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. *NOTE: 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> quote site setcos 4
```

2.1.10. Staging a File - stage

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

```
quote site stage <file>
```

where,

file is the name of the file.

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

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

2.1.11. Creating a Symbolic Link - symlink

symlink is used to create a symbolic link.

```
quote site symlink <path/file> <link>
```

where,

path/file refers to the destination

link refers to the local filename.

Example: The following may be entered to create a link named sys_passwd in the local directory pointing to /etc/passwd.

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

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

2.1.12. Setting the Desire Wait Options (for Migrated Files) - wait

wait is used to notify the HPSS PFTP Daemon :

```
quote site wait <option>
```

where,

option is one of the following values:

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

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

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

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

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

The following table describes the behavior the customer should expect from FTP when issuing the stage/wait commands. Only Classes of Service utilizing the "Stage on Open Background" option will exhibit predictable results. Refer to the *HPSS Installation Guide*, Section 3.10.3: *Class of Service* for more information.

Table 1: Stage/Wait Behavior

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

2.1.13. Changing the Default UMASK

umask is used to change the default umask.

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

Example:

```
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 your Support Representative for more guidance. PFTP supports the FTP command set plus some additional commands. Refer to *Section 3.2 – Additional HPSS 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 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; e.g. 1MB.
-c	Sets "Child" mode. This provides the ability to "emulate" a tty and interactive mode when executing the client in a "batch" mode.
-d	The standard FTP debug specification.
-e	Sets "Echo" mode. When running the client in batch mode, this causes the client to echo each command into the output file providing a helpful record of commands interleaved with the return messages.
-g	Disables the expansion of metacharacters in file names. Interpreting metacharacters can be referred to as expanding (sometimes called globbing) a file name. See the glob subcommand.
-h	Specifies to use the original HPSS protocol, PDATA_AND_MOVER, regardless of the default specified in the HPSS.conf file
-i	Turns off interactive prompting during multiple file transfers. See the prompt, mget, mput, and mdelete subcommands for descriptions of prompting during multiple file transfers.
-k	Kerberos ONLY option to specify an alternate Kerberos Realm for the PFTP Daemon.
-m	This argument will enable multinode processing. By default, multinode processing is disabled. Multinode will be ignored if NO multinode specification for this client / daemon pair is specified in the HPSS.conf file.
-n	Prevents an automatic login on the initial connection. Otherwise, the ftp command searches for a $HOME/.netrc entry that describes the login and initialization process for the remote host. See the user subcommand.
-p	Specifies to use the HPSS protocol (PDATA_ONLY) for parallel transfers regardless of the default in the HPSS.conf file.
-t	The standard FTP trace specification.
-v	Displays all the responses from the remote server and provides data transfer statistics. This display mode is the default when the output of the ftp command is to a terminal, such as the console or a display.
-w	This argument will set the pwidth. The pwidth value must be specified immediately following this argument.

-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; e.g. 1MB.
-C	Sets the default Class of Service (COS) for the session. The argument is a valid string representation of a decimal COS. COS names are NOT accepted.
-P	Specifies to use 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 250GB (subject to change) is compiled in. Results may be smaller than the specified value based on a number of external HPSS constraints. The default is 1.5GB. StringSize is the size specification in the format: Digit(s)Magnitude; e.g., 200GB.
-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 7.4 **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 your site representative 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 Parallel FTP Daemon using Kerberos credentials for authentication. NOTE: 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. *NOTE: Incompatibilities may exist between the HPSS (GSS) Parallel FTP Client and Daemon and the Kerberos-based FTP features provided by IBM with AIX 4.x and/or AIX 5.x. This has been reported in the past and was outside the jurisdiction of the HPSS project. The HPSS Parallel FTP Clients and Daemon **are** compatible with the MIT GSS-based FTP processes.*

As a courtesy to HPSS customers, the Parallel FTP Client code is available for compilation at customer sites upon request on non-HPSS platforms, subject to any legal/licensing restrictions. Contact your site representative for details. This explicitly denies any support requirement on IBM or the HPSS Development/Support personnel for any modifications made by the customer.

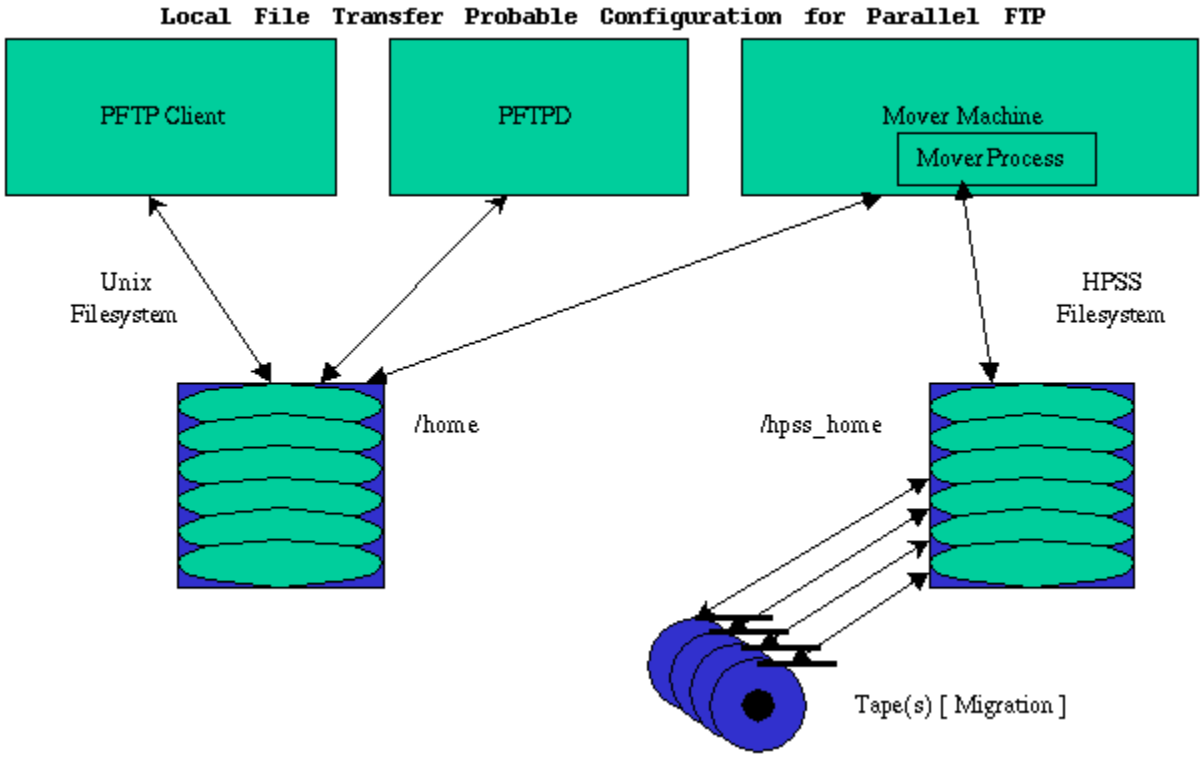
The HPSS PFTP Client has been successfully compiled on: Cray UNICOS [discontinued], Hewlett-Packard HP-UX [discontinued], Silicon Graphics IRIX [discontinued], Sun Solaris (Sparc and Intel), Intel Teraflop[discontinued], NEC[discontinued], IA64 Linux, Linux Intel (RHEL 4.0 & 5.0), SUSE 9.x for AMD64, Compaq Alpha[discontinued], and 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/software components are the responsibility of the remote site. These sites will be asked to share their ports with the HPSS development team (and other HPSS facilities); however, neither IBM nor the HPSS Development Team accepts any obligation to incorporate any hardware/software ports into the distribution source. No site specific features (local mods) added to the Parallel FTP client by customer sites will be incorporated into the PFTP client without the appropriate modification to the HPSS license agreement.

Note: 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).



The multinode option supports spawning the client processes across multiple machines / nodes and/or multiple interfaces on the remote machines / nodes. This Multinode option may be beneficial on processors which support shared file systems, such as GPFS on the IBM SP. ***NOTE: if multinode is used in a non-shared file system, the multinode file transfer to the client will be spread across multiple, separate files, which is not the desired 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 of 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 your site representative 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 the everybody and writable by root ONLY for security reasons.

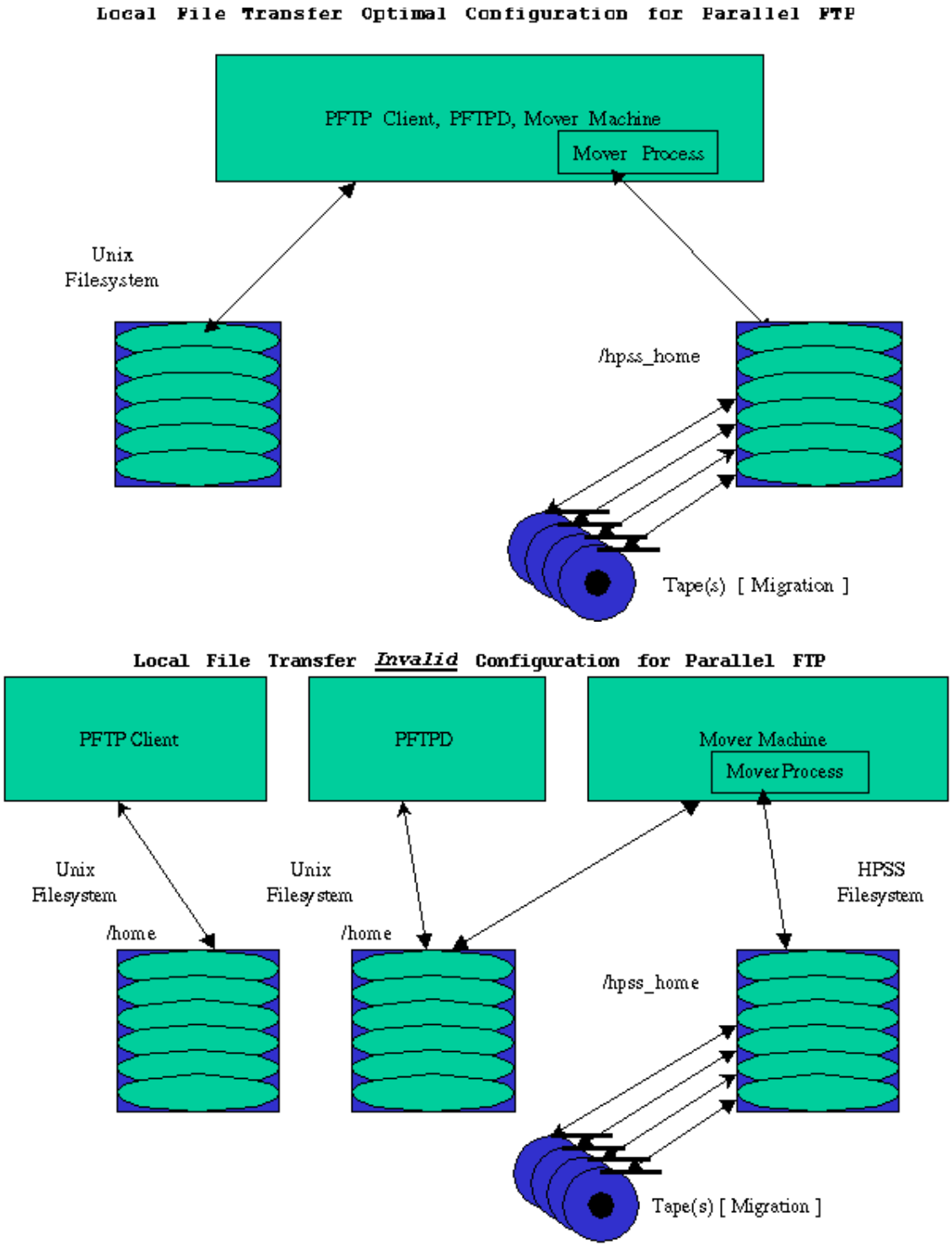
3.1.1.1. HPSS.conf File

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, contact your HPSS Administrator. This file is constantly being updated with additional features/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/destination file are "globally available" to the mover(s) and the PFTP client processes (e.g., 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:


Local File Transfer Optimal Configuration for Parallel FTP
PFTP Client, PFTPD, Mover Machine
Mover Process
Unix
Filesystem
/hpss_home
Tape(s) [Migration]
Local File Transfer Invalid Configuration for Parallel FTP
PFTP Client
PFTPD
Mover Machine
Mover Process
Unix
Filesystem
Unix
Filesystem
HPSS
Filesystem
/home
/home
/hpss_home
Tape(s) [Migration]

3.2. PFTP Site (and Quote) Commands

Several additional Site commands are available with PFTP.

Note: 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/Setting HPSS ACLs

acl is used to list/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/Setting Buffer Sizes [Grid FTP]

bufsize is used to determine/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 Parallel FTP Server is used to transfer files to the Client. **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

Minfo is used to determine which files in a directory are newer than a specified date. Minfo 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 and

path refers to the a 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 mover(s). Similarly, the quote wtrmt command is used to time the performance of writing data to the HPSS media. **NOT i**mplemented for HPSS PFTP Server!

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

NOTE: 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 (abbreviations) are supported by PFTP:

- pappend (papp)
- pput, mpput (ppu, mpp)
- pget, mpget (pge, mpg)
- lfappend (lfa)
- lfput, mlfput (lfp, mlfp)
- lfget, mlfget (lfg, mlf)
- setpwidth (setpw)
- setpblocksize (setpb)
- multimode (multi)
- autoparallel (autop)
- getprot (getp)
- gettuningparms (gettun)
- pdata (pdata)
- pdatapush (pdatap)
- pmover (pmov)
- setsockbufsize (sets)
- setxferbufsize (setx)

Pipes are NOT supported for the pget command. Pipes are supported by the pput command if appropriate configuration is performed. ***The use of pipes is strongly discouraged.***

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 7.4 Parallel FTP Daemon on fire
#               coming from fire.clearlake.ibm.com
#

220 fire FTP server (HPSS 7.4 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)

```
Parallel block size set to 1048576.
Connected to fire.clearlake.ibm.com.
220-#
#    HPSS 7.4 Parallel FTP Daemon on fire
#               coming from fire.clearlake.ibm.com
#

220 fire FTP server (HPSS 7.4 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.

Notes

none.

Examples

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

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

3.3.3. 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 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 amount of data transferred and any error conditions.

Error conditions

Connection Failures - data transfer connection malfunction.

Network Failures - data transfer malfunction.

Allocation Failures - no space on remote machine for file.

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

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

See also

RFC-0959.

Notes

none.

Examples

1. Transfer local file testfile to the user's HPSS home directory.
2. **`ftp> pput testfile`**
3. Transfer local file testfile to HPSS file prod1 in the current working directory.
4. **`ftp> pput testfile prod1`**
5. Transfer 1MB from offset 1MB of local file testfile to offset 0 of HPSS file /home/bob/prod1.
   `ftp> pput -l 1048576 -r 0 -s 1048576 testfile /home/bob/prod1.`

6. 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`

Parallel Pipes are NOT supported! The pipe facility requires explicit configuration. This should be avoided if possible.

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.

Notes

none.

Examples

1. Transfer all local files in the current directory to the user's HPSS home directory.
2. `ftp> mpput *`
3. Transfer all local files which begin with test in directory /usr/bob to the user's HPSS home directory.
4. `ftp> lcd /usr/bob`
5. `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 transfer. The *local_offset*, *remote_offset*, and *size* may be specified using a decimal and magnitude representation string. See Section 2.1.10 for use of this notation.

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

Parameters

-r *remote_offset* - Optional byte offset where 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.

Notes

none.

Examples

1. Transfer HPSS file /home/bob/prod1 to the user's local directory.
2. **`ftp> pget /home/bob/prod1`**
3. Transfer HPSS file prod1 in the current working directory to local file testfile1.
4. **`ftp> pget prod1 testfile1`**
5. Transfer 1MB from offset 0 of HPSS file /home/bob/prod1 to offset 1048576 of local file testfile.
6. **`ftp> pget -r 0 -l 1048576 -s 1048576 /home/bob/testfile1 testfile`**
7. 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 -"`
8. 
   Parallel Pipes are NOT supported for the pget command! Avoid the use of pipes if possible.

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.

Notes

none.

Examples

1. Transfer all files in HPSS directory /home/bob to the user's local directory.
```
ftp> cd /home/bob
ftp> mpget *
```

2. Transfer all HPSS files which begin with test in directory /home/bob to the user's local directory.
```
ftp> cd /home/bob
ftp> mpget test*
```

3.3.7. Local File append - lfappend

Synopsis

lfappend *local_file [remote_file]*

Description

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

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

Parameters

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.

Notes

none

Examples

1. Append "globally available" file testfile to the same file name in the user's HPSS home directory.
   `ftp> lfappend testfile`
2. . . . (Information returned by the PFTP Daemon)
3. **`ftp>`**
4. Append "globally available" file testfile to HPSS file prod1 in the current working directory.
   `ftp> lfappend testfile prod1`
5. . . . (Information 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 Parallel FTP Client command used to transfer a "globally available" file into HPSS using the "parallel" protocols. IF the file is not available to the mover(s) machine (NOT "globally available") the transfer will fail because the mover(s) will NOT be able to locate the desired file. The difference between lfput and pput is that the mover(s) involved in the transfer will not use the network to move the data providing improved performance. The mover(s) machine MUST be correctly configured. The file "/var/hpss/etc/hpss_mvr_localfilepath.conf" MUST exist on each "local file" aware mover machine and MUST contain entries specifying which directories are eligible for "local file" transport.

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

Parameters

-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. (MUST be available to mover(s) 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.

Notes

none.

Examples

1. Transfer local file testfile in the current working directory of the client to the user's HPSS home directory.
   `ftp> cd ~`

2. . . . (Information returned by the PFTP Daemon)
   `ftp> lfput testfile`

3. . . . (Information returned by the PFTP Daemon)
   `ftp>`

4. 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`

5. . . . (Information returned by the PFTP Daemon)
   `ftp>`

6. Transfer 1MB from offset 1MB of local file testfile in the current working directory of the client to offset 0 of HPSS file /home/bob/prod1 with a new name testfile2..
   `ftp> lfput -l 1048576 -r 0 -s 1048576 testfile /home/bob/prod1/testfile2`

7. . . . (Information 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 Parallel FTP Client command used to transfer an HPSS file into a "globally available" file using the "parallel" protocols. IF the current working directory or the specified directory is not available to the mover(s) machine (NOT "globally available") the transfer will fail because the mover(s) will NOT be able

to locate the desired file. The difference between lfget and pget is that the mover(s) involved in the transfer, will not use the network to move the data providing improved performance. The mover(s) machine MUST be correctly configured. The file: /var/hpss/etc/ hpss_mvr_localfilepath.conf MUST exist on each "local file" aware mover machine and MUST contain entries specifying which directories are eligible for "local file" transport.

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

Notes

none.

Examples

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

2. . . . (Information returned by the PFTP Daemon)
   ftp>

3. Transfer the HPSS file prod1 in the current working directory to local file testfile1.
   ftp> lfget prod1 testfile1

4. . . . (Information returned by the PFTP Daemon)
   ftp>

5. Transfer the HPSS file testfile into the "globally available" directory /home/bob renaming the file to testfile.
   ftp> lfget prod1 /home/bob/testfile1 testfile

6. . . . (Information returned by the PFTP Daemon)
   ftp>

7. Transfer 1MB from offset 0 of HPSS file /home/bob/prod1 to offset 1048576 of local file testfile.
   ftp> lfget -r 0 -l 1048576 -s 1048576 /home/bob/testfile1 testfile

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

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

Parameters

local_files - Identification of the files to transfer on the local machine.

Return strings

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

Error conditions

Connection Failures - data transfer connection malfunction.

Network Failures - data transfer malfunction.

Allocation Failures - no space on remote machine for file.

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

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

See also

RFC-0959.

Notes

none.

Examples

1. Transfer all "globally available" files in the current directory to the user's HPSS home directory.
   ```
   ftp> cd ~
   ```
2. . . . (Information returned by the PFTP Daemon)
   ```
   ftp> prompt(<== Toggles File Prompting)
   ```
3. . . .
   ```
   ftp> mlfput *
   ```
4. . . . (Information returned by the PFTP Daemon)
   ```
   ftp>
   ```
5. Transfer all "globally available" files which begin with test in directory /usr/bob to the user's HPSS home directory.
   ```
   ftp> cd ~
   ```
6. . . . (Information returned by the PFTP Daemon)

```
ftp> mlfput /usr/bob/test*
```

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

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

Parameters

remote_files - Identification of the files to transfer from the remote (HPSS) host.

Return strings

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

Error conditions

Connection Failures - data transfer connection malfunction.

Network Failures - data transfer malfunction.

Allocation Failures - no space on remote machine for file.

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

- -5 - an I/O error occurred.

See also

RFC-0959.

Notes

none.

Examples

1. Transfer all files in HPSS directory /home/bob to the "globally available" current working directory
   ```
   ftp> mlfget /home/bob/*
   ```

2. . . . (Information returned by the PFTP Daemon)
   ```
   ftp>
   ```

3. Transfer all HPSS files which begin with test in directory /home/bob to the "globally available" current working directory.
   ```
   ftp> mlfget /home/bob/test*
   ```

4. . . . (Information returned by the PFTP Daemon)
   ```
   ftp>
   ```

3.3.12. Specify transfer stripe width - setpwidth

Synopsis

setpwidth *stripe_width*

Description

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

Parameters

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

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

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

Enter the **lshpss -cos** command to list Class of Service information. From the entry with the *CID* value equal to your Class of Service ID, locate the *HID* (hierarchy ID) field.

Enter the **lshpss -hier** command to list hierarchy information. From the entry matching the *HID* value above, locate the *StorageClassID* field.

Enter the **lshpss -sc** command to list the storage class information. From the entry matching the *StorageClassID* value above, locate the *W* field. This is the stripe width for the storage class of the file.

Return strings

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

Error conditions

"Bad width value [stripe width]."

See also

RFC-0959.

Notes

none.

Example

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

3.3.13. 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 2.1.10 for use of this notation. The maximum blocksize is 16mb.

Parameters

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

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

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

Return strings

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

Error conditions

"Bad block size value [block size]."

See also

RFC-0959.

Notes

none.

Example

1. Set the transfer block size to 8 MB.
   ```
   ftp> setpblocksize 8388608
   ```
   or
2. ```
   ftp> setpblocksize 8MB
   ```

3.3.14. Multinode Enable/Disable - multinode

Synopsis

multinode

Description

The **multinode** command is used to enable/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.”
   ```

See also

None

3.3.15. Autoparallel Enable/Disable - autoparallel

Synopsis

autoparallel

Description

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

Parameters

None.

Return string

Automatic Substitution of Parallel Commands Disabled

Daemon supports Parallel Features - Auto-Parallel Substitution Enabled

Error conditions

?Invalid command

See also

HPSS.conf(7)

Examples

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

or

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

3.3.16. 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

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

3.3.17. 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 conditionsWarning Preceding without HPSS.conf (-2) (Observed at Login Time)

HPSS.conf parsing errors.

See also

None.

Example

```
ftp> gettun
Effective Tuning parameters from saux22 to sair031
     Using PDATA_AND_MOVER protocol
     Using 4.1 Protocol
     Parallel Transfer Size = 2147483647
     Transfer Buffer Size = 16777216
     Parallel Block Size = 262144


     Parallel Network Width = 1
     No Interfaces Found (ret_code = -2)
     Using Default Interface for 1 stripes(s)
     Multinode is disabled
```

or

```
     Multinode Enabled:
     Processing the  multinode list, please wait.....
     Using 1 remote node(s) from the following:
     Control Interface ==> Data Interface:
     water ==> water

     Using Network Options
      Using "Default" destination characteristics
     PdataSockBufSize = 1048576 based on user input
     recv_socksize = send_socksize = 1048576 based on
                                     PdataSockBufSize
     Writesize = 524288
```

or

```
     recv_socksize = 262144 based on Network Options
     send_socksize = 262144 based on Network Options
     PdataSockBufSize = 262144 based on send_socksize

     RFC1323 is turned on
     TCPNoDelay is turned on
```

NOTE: 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

3.3.18. 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>
```

3.3.19. 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>
```

3.3.20. 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

ftp> pmover

Server does NOT support command ==> Older Server?

?Invalid command ==> Older Client?

See also

HPSS.conf(7)

Examples

1. Set PDATA_AND_MOVER protocol

```
ftp> pmover
215 Parallel protocol is PDATA_AND_MOVER
```

```
ftp>
```

3.3.21. Set the Socket Buffer Size - setsock

Synopsis

setsock *SizeString*

Description

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

Parameters

SizeString; e.g., "1MB"

Return strings

Socket Buffer Size = 1048576.

Error conditions

PdataSockBufSize reset equal or below sb_max (1048576)

See also

None.

Example

1. Set Socket Buffer Size
```
ftp> setsock 4mb (Above system max)
PdataSockBufSize reset equal or below sb_max (1048576)
ftp>
```

2. Set Socket Buffer Size
```
ftp> setsock 512kb
ftp>
```

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

3.3.22. Set the Transfer Buffer Size - setxfer

Synopsis

setxferbufsize *SizeString*

Description

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

Parameters

SizeString; e.g., "4MB"

Return strings

PdataBufferSize = 4194304

Error conditions

?Invalid command ==> Old Client?

See also

None.

Example

1. Set Transfer Buffer Size
```
ftp> setxfer 40MB (Max is 32MB)
ftp>
```

2. Display effective Transfer Buffer Size (no argument)
```
ftp> setxfer
Socket Buffer Size = 33554432.
ftp>
```

Chapter 4. Linux Virtual File System (VFS)

This chapter specifies the HPSS VFS interface. VFS is supported from the Linux platforms identified in the *HPSS Management Guide* section 14.3.2.

4.1. VFS Interface

The HPSS VFS interface provides a POSIX file system interface to HPSS. The implementation is composed of two components: vfs kernel extension and a daemon process. The vfs kernel extension implements the Linux vfs layer for a file system, therefore providing the POSIX file system interface. The daemon process is the means to communicate with HPSS using the HPSS Client API.

POSIX system calls are first handed to the HPSS VFS kernel extension, then transferred to the daemon process, the daemon process converts the request into an HPSS client API call to the remote HPSS Core Server. Returned information from the HPSS client API call is then sent from the daemon to the HPSS VFS kernel extension, then returned back to the application that submitted the system call. The HPSS VFS kernel extension communicates directly with movers when sending and receiving file data.

Performance characteristics of the HPSS VFS interface should be comparable to the other HPSS interfaces (ftp, pftp, client api). A slightly slower response may be measured do to POSIX requests going through the kernel vfs layer, then to the daemon process, then to HPSS using the client API and back. There are many mount point options that can be used to optimize the communication with HPSS (see the HPSS Management Guide for further details). For instance, the HPSS VFS can be configured to take advantage of multiple network interface cards, transferring data in parallel.

The HPSS Client API does not support the file or file region locking interfaces (e.g., fcntl(2) & flock(2)) found in most Unix operating systems. As a result of this the VFS interface does not support locking across multiple VFS mount points. It is the responsibility of the end users and end user applications to handle any necessary serialization outside of the VFS interface.

Beyond the POSIX interface, there are a set of ioctl command options to provide some HPSS centric control over how HPSS manages the individual file (see *HPSS Programmer's Reference*).

4.2. Linux Extended Attributes

The HPSS VFS interface supports a subset of the Linux extended attributes for the system, user and security name spaces. For additional details on Linux extended attributes see the man pages for attr(5).

4.2.1. USER and SECURITY Attributes:

- The HPSS VFS implementation supports only Text values. A possible method to store binary values would be to convert them to text, such as using a base64 encoding.
- The maximum attribute name length is 250 bytes.
- The maximum attribute value length of an individual attribute is 983 bytes.

- The maximum name and value length maximums are independent of each other.
- The internal implementation within HPSS uses XML formating to store the attribute name and value. The following characters will be internally converted within HPSS VFS to XML escape strings. This will reduce the maximum name or value length by the following per each occurrence:
  - & reduces by 4
  - < reduces by 3
  - > reduces by 3
  - " reduces by 5
  - \ reduces by 5
- Do to being a text based implementation, a NULL terminator will be added to the value if one is not originally provided. When a NULL terminator is not provided, the value length is increased by one byte.
- The maximum capacity (number attributes, aggregate value total) for extended attributes is limited by the resources configured for the HPSS User Defined Attribute capability. See your system administrator to determine how the HPSS User Defined Attributes is configured for your system.
- The maximum aggregate size returned from a list attributes for user and security is 354 bytes. This is the sum of all the user and security attribute name and value lengths. When the aggregate is larger than this maximum, the listxattr() system call returns no user or security name space HPSS extended attributes, however the system namespace attributes are returned.
- When deleting an attribute that doesn't exist, the return code to the setxattr() system call is zero instead of ENOATTR.

4.2.2. SYSTEM Attributes:

System namespace attributes are specific to each Linux vfs implementation. The following HPSS VFS system attributes are available and are retrieved using the HPSS Client API hpss_FileGetXAttributes() or hpss_FileGetAttributes(), therefore they are not subject to the above limitations for user and security attributes:

- account – read / write. Returned from hpss_FileGetAttributes()
- bitfile – read only. BFID for the file from original file open call.
- realm – read only. Returned from hpss_FileGetAttributes()
- comment – read / write. Returned from hpss_FileGetAttributes()
- cos – read / write. Returned from hpss_FileGetAttributes()
- family – read / write. Returned from hpss_FileGetAttributes()
- fileset – read only. Returned from hpss_FileGetAttributes()
- opens – read only. Returned from hpss_FileGetAttributes()
- reads – read only. Returned from hpss_FileGetAttributes()
- writes – read only. Returned from hpss_FileGetAttributes()

- optimum – read only. This value is a combination of fields returned from hpss_FileGetXAttributes(). The format is XXX:YYY:ZZZ where XXX is the optimum access size, YYY is the stripe width, and ZZZ is the stripe length.
- level - read only. This value is a combination of fields returned from hpss_FileGetXAttributes(). For each level in the COS Hierarchy where data resides for the file, an entry is displayed.
  - Disk levels have the format: AAA.BBB.CCC.DDD.EEE.FFF where
    - AAA is the storage class level
    - BBB is the type of storage (disk)
    - CCC is the amount of data at the level
    - DDD is the stripe length
    - EEE is the stripe width
    - FFF is the optimum access size
  - Tape levels have the format: AAA:BBB:CCC:DDD:EEE:FFF:GGG:HHH:[III...JJJ] where
    - AAA is the storage class level
    - BBB is the type of storage (tape)
    - CCC is the amount of data at the level
    - DDD is the stripe length
    - EEE is the stripe width
    - FFF is the optimum access size
    - for each VV:
      - GGG is the bytes on VV
      - HHH is the position on VV
      - [III...JJJ] are each of the PV for the VV
  - Level information is separated by semicolons.

Appendix A. Glossary of Terms and Acronyms

ACI	Automatic Media Library Client Interface
ACL	Access Control List
ACSLS	Automated Cartridge System Library Software (Science Technology Corporation)
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.
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.
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.
bar code	An array of rectangular bars and spaces in a predetermined pattern which represent alphanumeric information in a machine readable format (e.g., 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.
BMUX	Block Multiplexer Channel
bytes between tape marks	The number of data bytes 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.
central log	The main repository of logged messages from all HPSS servers enabled to send messages to the Log Client. The Log Client forwards messages to the Log Daemon for recording in the central log.
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
Core Server	An HPSS server which manages the namespace 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 sub-system uses exactly one Core Server.
daemon	A UNIX program that runs continuously in the background.
DB2	A relational database system, a product of IBM Corporation, used by HPSS to store and manage HPSS system metadata.

debug	A log record type used to report internal events that can be helpful in troubleshooting the system.
DEC	Digital Equipment Corporation.
delog	The process of extraction, formatting, and outputting HPSS central log records. This process is obsolete in 7.4 and later versions of HPSS. The central log is now recorded as flat 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
drive	A physical piece of hardware capable of reading and/or writing mounted cartridges. The terms device and drive are often used interchangeably.
EFS	External File System
ERA	Extended Registry Attribute
ESCON	Enterprise System Connection
event	A log record type used to report informational messages (e.g., subsystem starting, 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.
FDDI	Fiber Distributed Data Interface
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
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.
halt	A forced shutdown of an HPSS server.
HDM	Shorthand for HPSS/DMAP.
HDP	High Density Passthrough

hierarchy	See Storage Hierarchy.
HIMF	HPSS Interim Metadata Format
HiPPI	High Performance Parallel Interface
HPSS	High Performance Storage System
HPSS-only fileset	An HPSS fileset that is not linked to an external filesystem (e.g., XFS).
IBM	International Business Machines Corporation
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
Imex	Import/Export
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	Input/Output Descriptor / Input/Output 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
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
LARC	Langley Research Center
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.

LCU	Library Control Unit
LDAP	Lightweight Directory Access Protocol
LLNL	Lawrence Livermore National Laboratory
LMCP	Library Manager Control Point
LMU	Library Management Unit
local log	An optional log file maintained by a Log Client. The local log contains formatted messages sent by HPSS servers running on the same node as the Log Client.
Location Server	An HPSS server that is used to help clients locate the appropriate Core Server and/or other HPSS server to use for a particular request.
Log Client	An HPSS server required on non-Mover nodes. The Log Client is responsible for delivering log messages to the local log file, the Log Daemon, SSM, and the syslog system as configured.
Log Daemon	An HPSS server responsible for writing log messages to the central log. There is one Log Daemon per HPSS installation.
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, status, debug, request, security, trace, or accounting.
logging service	An HPSS infrastructure service consisting of a central Log Daemon, one or more Log Clients, 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.
logging service processes	The Log Client and Log Daemon.
LRU	Least Recently Used
LS	Location Server
LTO	Linear Tape-Open. A half-inch open tape technology developed by IBM, HP and Seagate.
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})

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 and/or 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/or HPSS namespaces.
Mover	An HPSS server that provides control of storage devices and data transfers within HPSS.
MPS	Migration/Purge Server
MRA	Media Recovery Archive
MSSRM	Mass Storage System Reference Model
MVR	Mover
NASA	National Aeronautics and Space Administration
Name Service	The portion of the Core Server that providesa 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
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, event, or status.
NS	HPSS Name Service
NSL	National Storage Laboratory

object	See Managed Object
ONC	Open Network Computing
ORNL	Oak Ridge National Laboratory
OSF	Open Software Foundation
OS/2	Operating System (multi-tasking, single user) used on the AMU controller PC
PB	Petabyte (2^{50})
PFTP	Parallel File Transfer Protocol
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

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
RMS	Removable Media Service
RPC	Remote Procedure Call
SCSI	Small Computer Systems Interface
security	A log record type used to report security related events (e.g., 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 (e.g., disk devices, memory buffers, network addresses).
SMC	SCSI Medium Changer
SMIT	System Management Interface Tool
SNL	Sandia National Laboratories
SOID	Storage Object ID. An internal HPSS storage object identifier that uniquely identifies a storage resource. The SOID contains an unique identifier for the object, and an unique identifier for the server that manages the object.

source	The set of origins from which data is received during a data transfer (e.g., disk devices, memory buffers, network addresses).
SP	Scalable Processor
SS	HPSS Storage Service
SSA	Serial Storage Architecture
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 the command line user interface provided by the hpssadm program.
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.
status	A log record type used to report progress for long running operations.
STK	Storage Technology Corporation
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 namespace 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 (i.e., stripe width).
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 (i.e., 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
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.
TTY	Teletypewriter
UDA	User-defined Attribute
UDP	User Datagram Protocol
UID	User Identifier
UPC	Universal Product Code
UUID	Universal Unique Identifier

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 Messages 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'Energie Atomique - Centre d'Etudes de Bruyères-le-Châtel**) for providing assistance with development of NFS V3 protocol support.