

EZSTORAGE

Table of Contents

Preface	3
Introduction	4
Overview	5
Storage Summary	6
STORAGE Interfaces	7
FTP: General File Transfers	7
NFT: Locally Enhanced Transfers	7
HSI: Enhanced or Concurrent File Transfer Sessions	8
Additional Interfaces	8
Accessing STORAGE	10
Copies in Storage	11
Using FTP	12
Basic FTP Commands	12
FTP Example	14
FTP Pitfalls (with Storage)	16
Using NFT	18
NFT Command Syntax	18
NFT Commands by Task	19
NFT Example	20
Using HSI	21
HSI Command Line	21
HSI FTP Compatibility	21
HSI Examples	23
Additional Resources	23
Sharing Stored Files	24
Using Storage Groups	24
Setting Stored-File Permissions by Group	25
Reading Shared Stored Files	26
Storage Assistance Tools	27
LSTORAGE (List Stored Files)	28
CHMODSTG (Change Storage Permissions)	30
CHGRPSTG (Change Storage Groups)	33
HTAR (Manage Stored File Collections)	36
Disclaimer	38
Keyword Index	39
Alphabetical List of Keywords	40
Date and Revisions	41

Preface

Scope: EZSTORAGE explains how to transfer files between machines where you work (mostly LC production machines) and LLNL's High Performance Storage System (or STORAGE, LC's central file-storage archive). Transferring files using FTP and NFT as well as connecting to STORAGE from various locations is discussed. EZSTORAGE also tells how to overcome the most common problems encountered when storing and retrieving your files, including sharing stored files. Three customized (LC-only) storage-assistance tools to manage and monitor the groups and permissions of your stored files are also introduced (LSTORAGE, CHMODSTG, and CHGRPSTG), along with a fourth local tool (HTAR) that supports the fast, efficient storage of very large archive (TAR-like library) files.

Availability: When the programs described here are limited by machine, those limits are included in their explanation. Otherwise, they run under all LC systems.

Consultant: For help contact the LC customer service and support hotline at 925-422-4531 (open e-mail: lc-hotline@llnl.gov, SCF e-mail: lc-hotline@pop.llnl.gov).

Printing: The print file for this document can be found at

OCF: <https://computing.llnl.gov/LCdocs/ezstorage/ezstorage.pdf>
SCF: http://www.llnl.gov/LCdocs/ezstorage/ezstorage_scf.pdf

Introduction

This manual provides a basic guide to effectively storing and archiving files from LC computers by using the High Performance Storage System (HPSS), also referred to herein as STORAGE. Its goal is to introduce relevant background information, describe storage interface options, and give an overview of the basic commands used for storage. The [HPSS Manual](http://www.llnl.gov/LCdocs/hpss) (URL: <http://www.llnl.gov/LCdocs/hpss>) provides a detailed discussion of the STORAGE system and its specialized features.

EZSTORAGE first provides an [overview](#) (page 5) of the strengths and weaknesses of LC's storage system, including the software [interfaces](#) (page 7) compatible with STORAGE, a summary of [STORAGE commands](#) (page 6), and [accessing STORAGE](#) (page 10) from various locations. The second section explains how to save files using [File Transfer Protocol](#) (page 12)(FTP). The third section explains how to save files using [Network File Transfer](#) (page 18) (NFT). The next section answers the common but unfortunately complex question of how to [share stored files](#) (page 24). Finally, a concluding section introduces three special [storage-assistance](#) (page 27) software tools to simplify managing your stored files, along with another local tool ([HTAR](#) (page 36)) that efficiently stores large archive (library) files or retrieves members from within them even while the archive itself remains stored.

Reference manuals are available to provide detailed technical instructions on the tools and techniques introduced in EZSTORAGE, including manuals on [FTP](#) (URL: <http://www.llnl.gov/LCdocs/ftp>), [NFT](#) (URL: <http://www.llnl.gov/LCdocs/nft>), [HPSS](#) (URL: <http://www.llnl.gov/LCdocs/hpss>), [HTAR](#) (URL: <http://www.llnl.gov/LCdocs/htar>), and the local [Firewall](#) (URL: <http://www.llnl.gov/LCdocs/firewall>). Additionally, the document [EZFILES](#) (URL: <http://www.llnl.gov/LCdocs/ezfiles>) is a basic guide for using local directories and for general file-handling software at LC. (Because "secure FTP," called SFTP, relies on a different server than standard FTP, you can *not* use SFTP to store files or retrieve stored files at LLNL.)

The Hierarchical Storage Interface (HSI) communicates with HPSS with a user-friendly interface that makes it easy to transfer files and manipulate files and directories using familiar UNIX-style commands. HSI supports recursion for most commands as well as CSH-style support for wildcard patterns and interactive command line and history mechanisms, among its many other features. For more information, see the [HSI User Manual](#) (URL: <http://www.llnl.gov/LCdocs/hsi>).

Overview

Reliable, massive, archival data storage is a crucial part of any effective high-performance computing environment. The good news is that LC has such storage (officially called the High Performance Storage System, HPSS). The bad news is that layers of historically complex, asymmetric, security-driven interface features can make using storage easy in some circumstances but almost impossible in others.

Although the actual disk and tape resources for storing files at LC are large and elaborate, the user interface is constrained to use the FTP daemons exclusively. This means that all your interactions with storage occur through FTP clients (or local alternatives to FTP clients, such as NFT, parallel FTP (PFTP), or HTAR). FTP is standardized and easy to learn, but for many typical storage tasks it is inept or profoundly inelegant (the [file-sharing](#) (page 24) section below provides an extended example). Because SFTP talks to a different, non-FTP daemon, you cannot store or retrieve files using SFTP.

Moving files to/from LC production machines, open or secure, is a mainstream storage mission, easy to perform and very reliable. Using file storage in this context avoids quotas on user home directories, avoids purges of files on temporary work disks, and provides virtually unlimited capacity for managing data or computational output. Transfer rates are fast, and FTP connections are very reliable. Customized FTP-daemon interfaces to handle special storage needs (such as NFT for persistent storage transfers or HTAR for very efficiently making large archives directly in storage) are available here, too.

Moving files to/from other LLNL machines is more complex. Features of special FTP clients, together with the need to protect unusual file formats during transfer to or from storage, may call for taking extra steps.

Finally, moving files to/from non-LLNL.GOV machines, such as computers at other sites or the workstations of distant ASC collaborators, is the most complicated of the three situations. It requires either using a two-stage process or running extra enabling software (such as VPN). This may involve running FTP twice, or using nonFTP transfers to an LC production machine before actually storing the files with FTP (run on an LC machine). The details, suggested in a [later section](#) (page 10), call for entire separate manuals to explain thoroughly, and nevertheless often change to reflect unsettled security goals.

Storage Summary

This section briefly summarizes the chief storage-system constraints and tells how to perform the most important file-storage tasks at LC. We suggest you save it for ready reference.

Storage System Constraints:

```
Largest allowed file size: 10 Tbyte (using FTP/NFT interface)
                          68 Gbyte/member      | (using HTAR
                          10 Tbyte/archive      | interface)
```

```
Longest file name: 1023 characters
  (With HTAR, longest entry name or soft link is 100 characters)
```

```
Problem characters in file names:
  Treated as file filters: ? * {a,b}
  Forbidden FIRST characters: # - | ~
  Forbidden in any position: * ? [ {
```

Commands for Common File-Storage Tasks:

TASK:	FTP	NFT
Connect to storage:	ftp storage	nft
Make storage directory:	mkdir <i>dr</i>	(same)
Change storage directories:	cd <i>dr</i>	(same)
Store a file:	put <i>fl</i>	(same)
Retrieve a stored file:	get <i>fl</i>	(same)
Retrieve from within a stored archive:	See HTAR	See HTAR
Delete a stored file:	delete <i>fl</i>	(same)
List stored files:	dir	(same)
Change permissions (to <i>nnn</i>):	quote site chmod <i>nnn fl</i>	chmod <i>nnn fl</i>
Change "class of service" (COS):	site setcos <i>nnn</i>	setcos <i>nnn</i>
Start migration of stored file from tape:	quote site stage <i>fl</i>	(none)
Control file overwriting:		
.....Prevent overwriting	(none)	noclobber
.....Allow overwriting	(default)	clobber
Can be run visually by using HOPPER:	yes	yes

See the section [below](#) (page 27) on additional, customized software tools available on LC production machines (tools such as LSTORAGE, CHMODSTG, CHGRPSTG, and HTAR) for handling special interactions with your stored files.

Storage Home Directories:

Regardless of their access software (FTP, NFT), LC users arrive at HPSS in their storage home directory. This always has a pathname of the form

`/users/u[00-54]/username`

where *username* is your LC login name (for example: `/users/u34/jsmith`). This basic directory structure supports customized division into subdirectories (for example, by invoking FTP's or NFT's MKDIR command) as well as access control of stored files by using [storage groups](#) (page 24).

STORAGE Interfaces

FTP: General File Transfers

FTP clients and (server) daemons are available on all LC production and special-purpose machines in both the open and secure environments. FTP is the standard interface to the LC archival file storage system (both open and secure). When you run FTP (on an OTP or Kerberos-passworded LC machine) with STORAGE as the target host, access is "preauthenticated" and you are NOT prompted for your password. Also, on all LC production machines (but not necessarily on other LC machines), a parallel FTP client (equivalent to PFTP) is now the default. All files that are 4 MBytes or larger automatically move to or from storage using parallel FTP.

A concise summary of how to use FTP commands and features to store files, with annotated examples, is found in the Using FTP (page 12) section of this document (along with a subsection on known pitfalls). For a detailed discussion of the user commands, software responses, and error codes for the FTP file-transfer utility consult the FTP Reference Manual. (URL: <http://www.llnl.gov/LCdocs/ftp>)

WARNING: Not all available FTP clients interact equally well with HPSS. If you work on LC Linux/CHAOS clusters, you have access to `/usr/kerberos/bin/ftp`, but you should always instead run `/usr/bin/ftp` to store files. Under some circumstances the former (but not the latter) client refuses to log you into HPSS or needlessly asks you to "please login with USER and PASS."

NFT: Locally Enhanced Transfers

NFT is available on all LC production machines (open and secure, but not on some special-purpose hosts) and is a locally developed file transfer tool. Although NFT uses standard FTP daemons to carry out its file transfers, it offers such enhanced features as:

- A special NFT server that preauthenticates all NFT transfers, so all NFT executions are passwordless.
- NFT elaborately tracks and numbers all transfers. It automatically persists if system problems delay storing any file, and it keeps detailed records of your file-storage successes and problems.
- Input from and output to files is easy, and NFT's command syntax (unlike FTP's) lends itself to practical use in scripts and batch jobs.
- Some NFT commands especially facilitate transfers to and from STORAGE (so some users regard NFT as primarily a file-storage rather than a general file-transfer tool). Also, NFT automatically "routes" storage-related file transfers to take advantage of fast, jumbo-frame network connections whenever they are available (especially helpful for transfers between the Lustre parallel file system and storage).

A concise, task-oriented summary of how to use NFT commands and features, with annotated typical examples is found in the Using NFT (page 18) section of this document. For a complete analysis of NFT syntax and special features, along with a thorough alphabetical command dictionary, consult the NFT Reference Manual (URL: <http://www.llnl.gov/LCdocs/nft>).

HSI: Enhanced or Concurrent File Transfer Sessions

HSI provides a UNIX shell-style interface to HPSS. Directories and files can be listed using the "ls" command, and traversing directories can be accomplished with the "cd" command.

When HSI is launched, it performs the following actions:

- Parses command line options.
- Reads startup files (the user's \$HOME/.hsirc, and the system-wide hsirc file that is optionally installed by the system administrator), if they exist. In general, most settings that are defined in the system-wide hsirc file can be overridden by the user's private .hsirc file.
- Authenticates using one of the mechanisms that were enabled when the application was compiled, such as kerberos or a username/password combination.

If your file-transfer needs specifically involve placing many files into or retrieving them from a remote archive (TAR-format library) file, consult the HTAR Reference Manual (URL: <http://www.llnl.gov/LCdocs/htar>) for another LC-designed, locally deployed tool tailored to efficiently managing large archives in storage or on any preauthenticated FTP server.

Additional Interfaces

GRAPHICAL INTERFACES:

In addition to FTP and NFT, there are also other local tools that use FTP daemons for file transfer to STORAGE through graphical interfaces. One for workstations is XDIR; one for Macintosh computers is Fetch. Another visual file-transfer interface called HOPPER is installed on all LC production machines, open and secure. With HOPPER, you use your mouse to select (CTRL-CLICK) files from displayed tables, then connect to a target machine using your choice (from a menu) of FTP (no default host) or NFT (where STORAGE is always the default). A third alternative, "Connect to Storage (HPSS)," opens an FTP connection directly to your top-level STORAGE directory by default (thus mimicking NFT). See the HOPPER (URL: <http://www.llnl.gov/LCdocs/ezoutput/index.jsp?show=s4.8>) section of the EZOUTPUT guide for concise HOPPER instructions.

ARCHIVE BUILDER:

A different local interface, called HTAR (page 36), has been designed at LC primarily to efficiently transfer very large archive (TAR-like library) files to and from STORAGE on LC production machines, or to extract member files from within still-stored archives (you can optionally use HTAR for similar *nonSTORAGE* transfers too). HOPPER can also serve as a visual controller for HTAR.

Note:

At LC, currently only STORAGE interfaces based on FTP daemons are supported (such as FTP, PFTP, NFT, HTAR, XDIR, and Fetch). And despite its name, SFTP relies on the SSHD2 daemon, not the standard FTP daemon, so you can NOT store or retrieve stored files at LLNL using SFTP as an interface. However, executing FTP on any LC production machine (but not necessarily on any other LC machines) is equivalent to executing PFTP (Parallel FTP) by default. Executing the PFTP client overtly gives you access to numerous special parallel-transfer commands (such as PPUT and PGET). These extra PFTP commands are unnecessary on LC production machines (where ordinary FTP performs parallel transfers to and from storage automatically), but if you use STORAGE at *other* ASC (tri-lab)

sites you may need to invoke them. Refer to the documentation available from the HPSS Collaboration Web site (URL: <http://www.hpss-collaboration.org/hpss/administrators/documentation.jsp>) for details about PFTP commands.

Accessing STORAGE

Accessing STORAGE is most easily done from an LC production machine. Offsite users will encounter difficulties in connecting to STORAGE because of interface limitations as well as intentional security barriers to easy use.

When onsite, NFT, FTP, and HSI can be used to transfer files to and from STORAGE. The NFT interface is supported only by LC machines and can only be used to transfer files between them. It is not possible to use NFT from a machine outside 134.n.n.n (including other llnl.gov machines and all onsite desktop machines).

Offsite users can only use FTP. However, LC's firewall totally blocks all FTP traffic from every host outside the llnl.gov domain. To transfer files from machines outside llnl.gov to any LC machine, outside-the-firewall users have three choices:

(1) Log on to an LC production machine, then execute FTP on that machine and connect back to the outside machine where the sought files reside, using GET to retrieve them. It also requires an FTP server (not just a client) running on the outside machine, a problem for some workstations. As a second stage, you must then run FTP on the llnl.gov machine again to transfer the files to storage.

(2) Run secure copy SCP (described in [EZOUTPUT](http://www.llnl.gov/LCdocs/ezoutput) (URL: <http://www.llnl.gov/LCdocs/ezoutput>)) instead of FTP to transfer files toward an open-network LC machine. You must have previously installed SSH on the outside machine (LC's firewall allows SSH and SCP traffic from any IPA-authenticated outside host). Since storage.llnl.gov has no SCP server, however, you must again run FTP on the LC machine as a second stage to actually transfer your files to storage.

(3) Before you run FTP on your outside-the-firewall machine, get, install, configure, and execute a Virtual Private Network (VPN) client on that machine. Contact the LC Hotline to see if you are authorized to run a VPN client for access to LLNL. A VPN client borrows an llnl.gov IP address for your machine while it runs, and LC has confirmed that if you run VPN and FTP together under Windows, you can directly transfer files to storage.llnl.gov from outside the firewall (no staging to an LC production machine is needed). You may encounter vendor-compatibility problems with various versions of Windows or with other operating systems. See LC's [Firewall and SSH Guide](http://www.llnl.gov/LCdocs/firewall) (URL: <http://www.llnl.gov/LCdocs/firewall>) for full instructions on the fairly complex process of getting and using VPN to enable FTP. (You cannot use SFTP through VPN to store or retrieve stored files.)

Note also that both the SSH and VPN choices (above) for reaching storage.llnl.gov from external machines are subject to a general 2-hour timeout. A Web-based reauthentication mechanism is described in the "Internet" section of [EZACCESS](http://www.llnl.gov/LCdocs/ezaccess/index.jsp?show=s2.2.2) (URL: <http://www.llnl.gov/LCdocs/ezaccess/index.jsp?show=s2.2.2>).

Detailed instructions for the choices mentioned above, as well as a concise but thorough SSH overview, including role, annotated setup steps, basic execute lines, and troubleshooting tips, are available in LC's [Firewall and SSH Guide](http://www.llnl.gov/LCdocs/firewall) (URL: <http://www.llnl.gov/LCdocs/firewall>).

Copies in Storage

Some files may be so important to your project that you want to store separate, duplicate copies on independent storage media (at LC, this means separate tape cartridges). LC's OCF and SCF storage systems offer such dual-copy storage using the "class of service" (COS) concept.

The storage server(s) assign to every incoming file a COS based on the file's size and the client that writes it:

- Files written with FTP or NFT that are smaller than 32 Mbytes are *automatically* assigned a COS that provides two separate copies on separate storage tapes. For these files you never need to request duplicate storage.
- Files written with FTP or NFT that are 32 Mbytes or larger are assigned a COS that stores only a single copy. For mission critical files in this category you can request dual-copy storage by using the FTP command

```
site setcos dualcopy
```

or the NFT command

```
setcos dualcopy
```

before you PUT the large file(s) into HPSS. NFT's DIR command (if used with the -h option) reports the current COS for already stored files (in output column 3).

- Files written with HTAR, regardless of their size, always get a default COS that stores only a single copy. For mission critical files written with HTAR you can request dual-copy storage by using the (uppercase) command

```
-Y dualcopy
```

on the HTAR execute line that creates your stored archive (this overrides the HTAR_COS environment variable).

For more COS technical details, consult the SETCOS section of LC's [HPSS Manual](http://www.llnl.gov/LCdocs/hpss/) (URL: <http://www.llnl.gov/LCdocs/hpss/>).

Using FTP

Basic FTP Commands

FTP is a widely used file-transfer utility because it supports transfers between any machines that recognize the TCP/IP protocols, even if they have different architectures or operating systems. You must, however, log in to the remote machine and transfer the files interactively using your (remote) password. (See [FTP Pitfalls](#) (page 16) below for a few known storage problems with FTP.)

Because of the need for fast, reliable file transfers to and from STORAGE (i.e., storage.llnl.gov), that host uses special FTP servers and other LC machines use special FTP clients that can preauthorize your FTP login to STORAGE (only), so that no password is requested. All LC production machines (IBM/AIX and Linux/CHAOS) offer passwordless (preauthenticated) FTP service to STORAGE. The usage is the same as standard FTP, except for omitting the password request. Note also that on production LC machines, NFT, which favors the STORAGE system in several ways, also offers passwordless file transfer to and from STORAGE (see the [NFT](#) (page 18) section for details). Furthermore, on all LC production machines (but not necessarily on other LC machines), a parallel FTP client (equivalent to PFTP) is now the default. See the [FTP Reference Manual](#) (URL: <http://www.llnl.gov/LCdocs/ftp>) for instructions on invoking a nondefault nonparallel FTP client, which is less verbose. Also, you can use [HOPPER](#) (page 8) to run FTP to STORAGE graphically.

Most FTP implementations support many commands, but not always the same ones. The standard FTP commands, with their syntax and error codes, are detailed in the [FTP Reference Manual](#) (URL: <http://www.llnl.gov/LCdocs/ftp>). The following FTP commands are the most commonly used ones for basic file transfer:

- `cd pathname` changes directories (on the remote machine) to the one specified by *pathname*. By default, FTP GETs files from and PUTs files to the home directory of the remote machine, so you must change directories with CD if you need to transfer them to or from somewhere else.
- `pwd` reports the current working directory's pathname on the remote machine (to confirm uses of CD).
- `dir` lists the names and attributes of files in the current working directory on the remote machine.
- `get remotefile [localfile]`
retrieves *remotefile* and places it in the current directory of the local machine (where you are running FTP). The incoming file is called *remotefile* by default, or called *localfile* if you specify a name. (Use [HTAR](#) (page 36) instead if you want to retrieve a member file from within a still-stored archive.)
- `mget filelist` generalizes the GET command to transfer all the files in *filelist*, a blank-delimited list of remote files to retrieve to the current directory (where you are running FTP). MGET accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding transfer.

- parallel** [LLNL only] enables parallel file transfers on LC production machines. But remember that parallel file transfers are already ON by default to or from STORAGE for all files over 4 Mbytes (so typing PARALLEL here just reports the stripe width and block size).
- put *localfile* [*remotefile*]**
copies *localfile* into the current working directory of the remote machine you have logged in to with FTP. The outwardly transferred file is called *localfile* by default, or called *remotefile* if you specify a name.
- mput *filelist*** generalizes the PUT command to transfer all the files in *filelist*, a blank-delimited list of local files to copy to the home directory of the remote machine that you logged in to with FTP. MPUT accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding transfer.
- delete *remotefile***
removes *remotefile* from the current working directory of the remote machine you have logged in to with FTP. Use DIR to confirm your deletion.
- mdelete *filelist*** generalizes the DELETE command to remove all the files in *filelist*, a blank-delimited list of remote files to delete from the current working directory on the remote machine. MDELETE accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding deletion. WARNING: see the known pitfall of using MDELETE with wildcards to delete files from the LC storage system ([subsection below](#) (page 16)).
- help [*command*]**
lists the commands supported by the implementation of FTP that you are running, or (with an argument) briefly describes one command.
- quit** closes your remote session and terminates FTP.

FTP Example

This annotated example shows a typical file transfer to and from STORAGE using FTP.

GOAL: To transfer files to and from STORAGE interactively using FTP (the default parallel FTP client). In this case, the local machine on which the user (JANE) executes the FTP client is ATLAS, and the remote machine that files are saved to and retrieved from is STORAGE.

STRATEGY: (1) The user runs FTP (on ATLAS) with storage.llnl.gov as the remote machine's domain name.
(2) Because STORAGE is a special destination, its FTP server preauthenticates JANE and asks for neither her user name nor her password (most other destinations ask for both). This dialog is more verbose than that for most FTP sites and automatically enables parallel file transfers.
(3) At the ftp> prompt, the user GETs file TEST5 (copies it from STORAGE to ATLAS).
(4) At the next ftp> prompt, the user PUTs file TABLE.DAT (copies it from ATLAS to STORAGE).
(5) When the file transfers are done and confirmed, the user QUITs FTP.

```
(1) ftp storage.llnl.gov
    Connected to toofast54.llnl.gov
    220-NOTICE TO USERS [long legal disclaimer here...]
    220 toofast15 FTP server (HPSS 7.1 PFTPD V1.1.1
        Tue Sep 8 14:06:03 PDT 2009) ready.

(2) Preauthenticated FTP to toofast54.llnl.gov as jane:
    232 GSSAPI user ... is authorized as jane@spectrum.llnl.gov
    230 User ... logged in as jane@spectrum.llnl.gov
    Remote system type is UNIX.
    Using binary mode to transfer files.

(3) ftp> get test5
    .
    .
    .
    226 Transfer complete.
    1827811 bytes received in 0.20 seconds (9.10 Mbytes/s)

(4) ftp> put table.dat
    .
    .
    .
    226 Transfer complete.
    7311244 bytes sent in 1.85 seconds (3.96 Mbytes/s)

(5) ftp> quit
    221 Goodbye.
```



You can work around this broad interpretation of FTP's MDELETE command by the LC storage server in several ways:

- Use as your MDELETE argument an explicit list of files instead of a file filter with metacharacters.
- Use a more restrictive filter (certainly more restrictive than * alone), carefully crafted to select only the files in the current working directory in storage (if your file-naming scheme allows).
- Use NFT (page 18) instead of FTP as your storage interface. NFT has no MDELETE command, and its separate storage server interprets DELETE * to remove only files in the current working directory (not in any child directories). You must invoke the -R suboption explicitly to make the NFT commands DELETE and RMDIR behave recursively.

CLIENT INTERACTION DIFFERENCES.

Not all available FTP clients interact equally well with HPSS. If you work on LC Linux/CHAOS clusters, you have access to /usr/kerberos/bin/ftp, but you should always instead run /usr/bin/ftp to store files. Under some circumstances the former (but not the latter) client refuses to log you into HPSS or needlessly asks you to "please login with USER and PASS."

Using NFT

NFT Command Syntax

NFT (unlike FTP) was designed for the LC environment and it has two special features that affect how you can use it.

First, all NFT file transfers involve not only the donor and the receiver machines you specify (overtly or by default), but also a third invisible machine running locally developed software dedicated to failure detection and recovery and NFT job tracking. If a problem prevents a file from being sent or received immediately, the software automatically remembers the request and persists in trying to complete it later, recording its results for you to verify if needed.

Second, NFT assumes that the remote host in all file transfers is the LC storage system (storage.llnl.gov) unless you specify otherwise. The existence of such a default remote host means that: (1) There are two types of NFT commands (one for general file transfers among any hosts that NFT serves, and one for those that do not accept NFT's usual host-specifying syntax because they default to storage transfers) and, (2) The syntax of NFT commands assumes local-to-storage transfers unless you specify otherwise. (You *can* use HOPPER (page 8) to run NFT as a controllee, but HOPPER's own "Connect to Storage (HPSS)" option runs FTP instead even though, like NFT, storage.llnl.gov is its default destination.)

NFT also has many special features that suit it for use in batch jobs and scripts (unlike FTP). The NFT Reference Manual (URL: <http://www.llnl.gov/LCdocs/nft>) explains those unusual features, while its basic commands pertaining to STORAGE are summarized here. Unlike FTP, NFT transfers involve no long legal preamble and no increased verbosity when connecting to a parallel FTP server such as STORAGE. Also unlike FTP, NFT automatically "routes" storage-related file transfers to take advantage of fast, jumbo-frame network connections whenever they are available. If you want to retrieve a member file from within a still-stored archive, use HTAR (page 36) instead.

Most NFT messages about *user* errors (as illustrated in the example (URL: <http://www.llnl.gov/LCdocs/ezstorage/index.jsp?show=s4.3>) dialog subsection below) are sequentially numbered along with other NFT responses and begin with the string "error":

```
n.0 error explanation
```

However, if HPSS storage is down for planned servicing, then attempts to store files using NFT yield a different error message with this special format:

```
[SCF|OCF] HPSS Storage is down for maintenance.  
*** Try again later.
```

Scripts that execute NFT should check for this special "three-star" error message and avoid needlessly resubmitting NFT jobs (storage requests) once it has been detected.

NFT Commands by Task

This chart shows the interactive NFT commands that perform the most common file-transfer tasks. For a detailed list of every NFT command, see the [Command Dictionary](http://www.llnl.gov/LCdocs/nft/index.jsp?show=s9) (URL: <http://www.llnl.gov/LCdocs/nft/index.jsp?show=s9>) in the NFT Reference Manual.

File-Transfer Task	NFT Command
GENERAL	
Change remote directory to <i>newdir</i> on <i>aaa</i>	<code>cd <i>aaa:newdir</i></code>
Change local directory to <i>newdir</i>	<code>cd <i>:newdir</i></code>
Change storage(*) directory to <i>newdir</i>	<code>cd <i>newdir</i></code>
List remote directory contents on <i>aaa</i>	<code>dir <i>aaa</i>:</code>
List local directory contents	<code>dir :</code>
List storage(*) directory contents	<code>dir</code>
Put (copy) local file <i>t1</i> to remote host <i>aaa</i> as <i>t2</i>	<code>cp <i>:t1 aaa:t2</i></code>
Get (copy) remote file <i>t3</i> from <i>aaa</i> as local file <i>t4</i>	<code>cp <i>aaa:t3 :t4</i></code>
Transfer file <i>t5</i> on <i>aaa</i> to file <i>t6</i> on <i>bbb</i> (both remote)	<code>cp <i>aaa:t5 bbb:t6</i></code>
Delete remote file <i>t3</i> from storage	<code>delete <i>t3</i></code>
STORAGE DEFAULTED(*)	
Store local file <i>t1</i>	<code>put <i>t1</i></code>
Store local file <i>t1</i> as <i>t2</i>	<code>put <i>t1 t2</i></code>
Retrieve from storage file <i>t3</i>	<code>get <i>t3</i></code>
Retrieve <i>t3</i> as local file <i>t4</i>	<code>get <i>t3 t4</i></code>
CONTROL OPTIONS	
Prevent all overwriting (default)	<code>noclobber</code>
Allow overwriting (for updates)	<code>clobber</code>
Start a log of NFT actions	<code>log <i>logfile</i></code>
Close the log file	<code>clog</code>
Change storage <u>class of service</u> to <i>nnn</i>	<code>setcos <i>nnn</i></code>
Change default remote host(*)	<code>open <i>host</i></code>
Terminate NFT	<code>quit (*)</code>

(*) You can change NFT's default remote host from storage to something else by using the OPEN command, but you should consult the [OPEN command section](http://www.llnl.gov/LCdocs/nft/index.jsp?show=s9.28) (URL: <http://www.llnl.gov/LCdocs/nft/index.jsp?show=s9.28>) of the NFT manual before you rely on it.

NFT Example

This annotated example shows typical file transfers using NFT.

GOAL: To transfer files from a secure LC machine to STORAGE without logging on to all of the machines, using NFT.

STRATEGY: (1) Start NFT. Notice that unlike FTP, you do not log on to any particular remote host to "open a connection."
(2) Use storage-defaulted command PUT to transfer file t1 from the client machine (where NFT runs) to storage.llnl.gov as file t2. Note that NO hosts are specified in this command because the default location is STORAGE.
(3) Try to retrieve file t2 from STORAGE to local file t1 using the storage-defaulted GET command. Because NFT's default environment is NOCLOBBER, this attempt fails (t1 already exists). You could use the CLOBBER option next, to allow this overwrite, or...
(4) Use GET to retrieve t2 from storage with no name change (and hence no overwriting of t1).

```
(1) nft
(2) nft>put t1 t2
4.0. 95 bytes sent in 1.0 seconds
(0.1 Kbytes/s) from /g/g0/jfk/t1 to ~/t2
(3) nft>get t2 t1
5.0. error. Cannot clobber existing sink.
/g/g0/jfk/t1
(4) nft>get t2
6.0. 95 bytes received in 1.8 seconds
(0.1 Kbytes/s) from ~/t2 to /g/g0/jfk/t2
nft>quit
```

Using HSI

HSI Command Line

The HSI command line has the following format:

```
hsi [options] [command [;command [;...]] ]
```

If an optional command-string is specified, then HSI is running in single execute line mode (also known as "one-liner" mode). In this mode, HSI will execute the command-string and then terminate. Multiple commands may be specified and separated by the semicolon (;) character. The command line may need to be enclosed in single or double quotes to protect it from expansion by the shell program that launches HSI.

You can also use HSI interactively by entering the command HSI without any arguments. Then simply enter HSI requests in response to the question-mark (?) prompt. Terminate the HSI session with the END command or one of its several aliases, such as BYE or QUIT.

Single-line execution is often used when HSI is run from within UNIX scripts and pipelines. An example of using HSI with two commands on the execute line might be:

```
hsi "cd /users/project1;ls -l *.c"
```

HSI FTP Compatibility

HSI supports several of the commonly used FTP commands, including DIR, GET, LS, MDELETE, MGET, PUT, MPUT, and PROMPT, with the following differences:

- The DIR command is an alias for LS in HSI. The LS command supports an extensive set of options for displaying files, including wildcard pattern-matching and the ability to recursively list a directory tree.
- The PUT and GET family of commands support recursion.
- There are "conditional" put and get commands (CPUT, CGET).
- The syntax for renaming local files when storing files to HPSS or retrieving files from HPSS is different than FTP. With HSI, the syntax is always *local_file* : *hpss_file*, and multiple such pairs may be specified on a single command line. With FTP, the local filename is specified first on a PUT command, and second on a GET command. For example, when using HSI to store the local file "file1" as HPSS file "hpss_file1" and then retrieve it back to the local file system as "file1.bak", the following commands could be used:

```
put file1 hpss_file  
get file1.bak : hpss_file1
```

With FTP, the following commands could be used:

```
put file1 hpss_file1  
get hpss_file1 file1.bak
```

- The "m" prefix is not needed for HSI commands; all commands that work with files accept multiple files on the command line. The "m" series of commands are intended to provide a measure of compatibility for FTP users.

HSI Examples

Save a "tar file" of C source programs and header files:

```
tar cf - *.*[ch] | hsi put - : source.tar
```

Note: the ":" operator that separates the local and HPSS path names must be surrounded by white space (one or more space characters)

Restore the tar file source saved above and extract all files:

```
hsi get - : source.tar | tar xf -
```

Get all files in the subdirectory subdira that begin with the letters "b" or "c" (surrounding the wildcard path in single quotes prevents shells on UNIX systems from processing the wildcard pattern):

```
hsi get 'subdira/[bc]*'
```

Save your local files that begin with the letter "c" (let the UNIX shell resolve the wildcard path pattern in terms of your local files by not enclosing it in quotes):

```
hsi put c*
```

Delete all files beginning with "m" and ending with 9101 (note that this is an interactive request, not a one-liner request, so the wildcard path does not need quotes to preserve it):

```
hsi <RETURN>  
? delete m*9101
```

Interactively delete all files beginning with H and ending with a digit, and ask for verification before deleting each such file.

```
hsi <RETURN>  
? mdel H*[0-9]
```

Interactively descend into the "Source" directory and move all files that end in ".h" into a sibling directory (i.e., a directory at the same level in the tree as "Source") named "Include":

```
hsi <RETURN>  
? cd Source  
? mv *.h ../Include
```

Additional Resources

For more information about HSI, see the [HSI User Manual](http://www.llnl.gov/LCdocs/hsi) (URL: <http://www.llnl.gov/LCdocs/hsi>).

Setting Stored-File Permissions by Group

Once you have the files you want to share and the name of a group to whom all sharing users belong (see the previous subsection), you can follow these steps, all involving (somewhat unusual) FTP commands, to enable the sharing of stored files:

(1) Open an FTP session to STORAGE.

All file-sharing arrangements require passing group and permission information to the storage system using the indirect mechanism that FTP provides for such nonstandard activity.

```
ftp storage
```

(2) Create a storage directory to hold the shared files.

In this example, the shared-files directory is called *share* and the shared file is called *share.code* (see also the figure at the start of the file-sharing section), but these can obviously be generalized as you need. In your FTP session type

```
mkdir share
```

(3) Assign your storage home directory to the share group.

If your default arrival directory in storage is `/users/u34/jfk` and if the storage group containing all the file-sharing users is *sgroup*, then use this indirect FTP command

```
quote site chgrp sgroup /users/u34/jfk
```

to associate the two. One side effect is that you cannot share with two different groups at once. (You can also change storage groups for any of these steps by using the special CHGRPSTG (page 33) tool, described in a later section.)

(4) Assign your file-sharing directory to the share group.

Because you made the share directory as a child of `/users/u34/jfk` in step (2), you can now associate it too with the file-sharing storage group *sgroup*:

```
quote site chgrp sgroup share
```

(5) Assign group permissions to the file-sharing directory.

To allow other members of storage group *sgroup* to read, write, and execute (list) the file(s) in the share directory, use this indirect FTP command

```
quote site chmod 775 share
```

to expand its default group permissions. (You can also change storage permissions for any of these steps by using the special CHMODSTG (page 30) tool, described in a later section.)

(6) Store the files to be shared.

If you move (CD) to the file-sharing directory and PUT the file(s) to be shared, they will lose their online permissions but they will arrive associated with the share group *sgroup*, which they inherit from the file-sharing directory:

```
cd share
put share.code
[more puts if there are more files to share]
```

(7) Assign group permissions to the file(s) to be shared.

Even if their online permissions allowed sharing by group, storing the file(s) erased those decisions. So as with step (5) above, you need to declare the availability of each file to the members of *sgroup*:

```
quote site chmod 775 share.code
```

Reading Shared Stored Files

After you have used the previous two subsections to enable others in storage group *sgroup* to share the file(s) in the share directory, they can follow these steps to retrieve those file(s):

```
ftp storage
cd /users/u34/jfk/share
get share.code
```

Note that impatient attempts to directly GET file `/users/u34/jfk/share.code` (while in another storage directory) may misleadingly fail with the message "no such file or directory."

Storage Assistance Tools

LC's production machines offer three public user-developed programs to handle three common storage tasks more conveniently than is possible with FTP or NFT. In fact, these storage assistance tools perform some helpful tasks (such as recursive changes on stored files) not possible with FTP (NFT offers a suboption, -R, that you can invoke to recursively change stored files).

These special storage tools and their roles are:

- lstorage** lists your storage directories and stored files in any of several formats, recursively if you request.
- chmodstg** changes the UNIX permissions on your storage directories or your stored files, recursively and symbolically if you request.
- chgrpstg** changes the (storage) group for your storage directories or your stored files (to enable file sharing), recursively if you request.

All are located in /usr/local/bin on the machines where they they have been installed (so most users can run them just by typing their names).

WARNING: Because all three storage-assistance tools are really Perl scripts, they yield very verbose and confusing error messages if you happen to run them when the LC storage system (either open or secure) is offline for maintenance.

In addition, LC provides a special-purpose front-end to parallel FTP that is customized to very efficiently store and retrieve large archive (TAR-format library) files. This combination file bundler and fast STORAGE interface is called HTAR. A short subsection below introduces HTAR's features and syntax, while LC's separate [HTAR Reference Manual](http://www.llnl.gov/LCdocs/htar) (URL: <http://www.llnl.gov/LCdocs/htar>) gives a thorough analysis of both good usage and known pitfalls. HTAR also offers the unique ability to retrieve a member file from within a still-stored archive, even without staging the archive from tape to disk in HPSS.

HTAR users may also benefit from familiarity with the HSI utility, which provides a user-friendly UNIX-style interface to HPSS, with a number of features, such as the ability to recursively store, retrieve and list entire trees with a single command. Additional HSI and HTAR information is available at <http://www.mgleicher.us> (URL: <http://www.mgleicher.us>).

LSTORAGE (List Stored Files)

EXECUTE LINE.

LSTORAGE lists your storage directories and the files that they contain. To run LSTORAGE on the LC production machines where it is installed, type

```
lstorage [options] [dirname]
```

By choice of LSTORAGE options you can specify output format (single or multiple columns), output scope (local or recursive), and level of detail (names only or other information too). The basic pattern for using LSTORAGE options is:

	Recursive	Nonrecursive

Single column	-lR, -j	-l
Multiple column	-R	default, -C

Because LSTORAGE runs noninteractively, redirecting its output to a file for later reuse is easy (e.g., `lstorage > outfile`).

DEFAULTS.

Without a specified directory, LSTORAGE reports on your top-level ("home") storage directory. Without options, LSTORAGE lists (only) the names of files and directories contained in the specified storage directory, in multiple columns. If you specify a space-delimited list of several target storage directories (all names relative to your home storage directory), LSTORAGE reports on each one in the order in which you listed them on the execute line.

SPECIAL BENEFITS.

LSTORAGE takes the place of using FTP's DIR or LS options. Unlike FTP, LSTORAGE avoids the long warning message, can make recursive reports, is easy to redirect, and can report on several storage directories at once.

TYPICAL USES.

```
lstorage -lR > storage.list
```

places into the file storage.list a detailed, recursive report on all of your storage directories and stored files (and their properties), starting with your "home" storage directory and working down the tree.

```
lstorage -j project2/admin
```

lists the names (only) of your storage directories, subdirectories, and stored files starting with the project2/admin directory and continuing recursively downward through the tree. The list is a single column indented at every new level to reveal nesting.

OPTIONS.

Scope options:

- a lists all directories and files, including those whose names begin with a dot(.).
NOTE: on the one hand, listing stored files such as .cshrc is default behavior for LSTORAGE even without invoking -a; on the other hand, even with -a invoked the list still omits the single and double dot (. and ..) entries that FTP's DIR reports.
- l lists in long format, with details on the permissions and groups for every storage directory or stored file covered in the report.
- R recursively includes all the children (subdirectories and stored files) of the directory specified on the execute line (compare with -j).

Format options:

- C (default) lists storage directories and stored files in multiple columns with entries sorted down the columns.
- j lists storage directories and stored files recursively (entails -R) in a single column with nesting revealed by extra indenting (names only).
- h displays the LSTORAGE help package (a brief list of options). Help cannot be combined with any other options.
- t *sss* sets the LSTORAGE timeout to *sss* seconds (default timeout is 300 seconds).

CHMODSTG (Change Storage Permissions)

EXECUTE LINE.

CHMODSTG changes the permissions on your storage directories or your stored files. To run CHMODSTG on the LC production machines where it is installed, type

```
chmodstg [options] [dirname]
```

By choice of CHMODSTG options you can specify the desired permissions for a specific storage directory, a specific stored file, all files in a directory, or (recursively) all children of a specific directory to all levels. You can also specify uninterrupted, noninteractive changes or instead request interactive prompting for your desired permissions and files (with optional report on each change made). The basic pattern for using CHMODSTG options is (all except -s can be combined with -R for recursive scope):

	Prompts		No prompt
	For perms only	For perms and files	
No reports	default, -f, -d, -s	-i	-F <i>perm</i> -D <i>perm</i>
Report results		-v or lstorage -l	-v or lstorage -l

DEFAULTS.

Without a specified directory, CHMODSTG acts on your top-level ("home") storage directory. Without permission-related options (e.g., `chmodstg -R dir1`), CHMODSTG prompts for your desired directory and file permissions and then changes both with no confirmation.

SPECIAL BENEFITS.

CHMODSTG takes the place of using FTP's QUOTE SITE CHMOD indirect command. Unlike FTP, CHMODSTG avoids the long warning message, can make recursive changes, and accepts symbolic rather than only octal permissions.

TYPICAL INTERACTIVE USES.

```
chmodstg -R project2/admin
```

announces that CHMODSTG will act recursively starting from the specified directory, prompts for your desired permissions on storage directories, prompts (separately) for your desired permissions on stored files, then changes the permissions without confirmation.

```
chmodstg -iR project2/admin
```

same as above (for -R), but also prompts for your yes/no choice for each directory and file processed.

chmodstg -ivR project2/admin

same as above (for -iR), but also reports the specific change made for every file (e.g., "changed from 750 to 700") as it occurs.

TYPICAL NONINTERACTIVE USES.

chmodstg -D775 project2/admin

assigns permission 775 to all subdirectories of the specified storage directory (use -s to change that directory itself), without prompting or confirmation.

chmodstg -D750 -F650 -R project2/admin

starts at the specified directory and recursively assigns 750 to every subdirectory and 650 to every stored file encountered as it works down the tree, without prompting or confirmation.

SPECIFYING PERMISSIONS:

CHMODSTG accepts permissions as either three-digit octal numbers (exactly three digits, no spaces) or as a comma-delimited list of symbolic triples (e.g., u+x,g-w) built up from the UNIX components [augo], [+ -], and [rwx]. (FTP accepts only the octal format.) Users unfamiliar with either style of specifying permissions can read a concise, overtly diagrammed summary of both in the "How to Specify Permissions" section of the EZFILES (URL: <http://www.llnl.gov/LCdocs/ezfiles>) basic guide.

OPTIONS.

Permission options:

-Fperm specifies (in either octal or symbolic format) the UNIX permissions *perm* to assign to every stored file (but not directories) that CHMODSTG treats during this run, as selected by other options. This disarms the file-permissions prompt.

-Dperm specifies (in either octal or symbolic format) the UNIX permissions *perm* to assign to every storage directory (but not stored files) that CHMODSTG treats during this run, as selected by other options. This disarms the directory-permissions prompt.

Scope options:

-f changes file permissions only (omits directories). CHMODSTG prompts you for the desired permissions. The default without -f or -d is to change both.

-d changes directory permissions only (omits files). CHMODSTG prompts you for the desired permissions. The default without -f or -d is to change both.

-s pathname changes permissions only for the one directory or file specified by its pathname (relative to your home storage directory). Using -s disables all other CHMODSTG options except -v, so CHMODSTG always prompts for your desired permissions even if you include -F or -D on the execute line.

-R recursively includes all the children (subdirectories and stored files) of the directory specified on the execute line. You can combine -R with other options (except -s) to further control CHMODSTG's scope of action.

Interaction options:

- i prompts for your yes/no confirmation for every directory or stored file that CHMODSTG tries to change (regardless of whether you also want prompting for desired permissions). Any response except YES is treated as NO; you can NOT supply different permissions for different files by using -i.
- v interactively reports the permission change made for every directory or stored file that CHMODSTG changes (e.g., "changed from 650 to 700"). You can combine -v with CHMODSTG's various prompting options, or use it for confirmations even without prompts.
- s is a scope option (see above) but always behaves interactively, even if you try to disable its prompts.
- h displays the CHMODSTG help package (a brief list of options). Help cannot be combined with any other options.

CHGRPSTG (Change Storage Groups)

EXECUTE LINE.

CHGRPSTG changes the group for your storage directories or your stored files. To run CHGRPSTG on the LC production machines where it is installed, type

```
chgrpstg [options] groupname [dirname]
```

There is no prompt or default for the desired *groupname*, which you must specify on every CHGRPSTG execute line. To discover your current groups, type the GROUPS command on LC production machine.

Most CHGRPSTG invocations run noninteractively, but you can request prompting or confirmatory reports, alone or together with recursive execution, by following this pattern of options:

	Recursive	Nonrecursive
Prompts and reports	-ivR	-iv
No prompts or reports	-R	default

DEFAULTS.

Without a specified directory, CHGRPSTG acts on your top-level ("home") storage directory. Without options, CHGRPSTG changes the group for one "layer" in your storage hierarchy (for every member of a specified directory but not the directory itself nor the children of its subdirectories). See the comparative example below.

SPECIAL BENEFITS.

CHGRPSTG takes the place of using FTP's QUOTE SITE CHGRP indirect command. Unlike FTP, CHGRPSTG avoids the long warning message, can make recursive changes, and can (optionally) treat just files or just directories at any level in your storage hierarchy.

TYPICAL INTERACTIVE USES.

```
chgrpstg -ivR newgrp project2/admin
```

announces that CHGRPSTG will act recursively starting from the specified directory, prompts for your yes/no choice for each directory and file processed, and reports the specific change made for every file (e.g., "changed from oldgrp to newgrp") as it occurs.

TYPICAL NONINTERACTIVE USES.

chgrpstg -s newgrp project2/admin

(change exactly one directory) changes the storage group only for the single directory specified. Note the different syntax from CHMODSTG (group name precedes pathname).

chgrpstg newgrp project2/admin

(change one "layer," the default) changes the storage group for all files and directories within the specified directory, but not for that directory itself nor for any children of its subdirectories.

chgrpstg -R newgrp project2/admin

(change all layers) changes the storage group for all files and directories within the specified directory, and also for all of its children working recursively down your storage hierarchy.

OPTIONS.

Scope options:

-f changes file groups only (omits directories). The default without **-f** or **-d** is to change both.

-d changes directory groups only (omits files). The default without **-f** or **-d** is to change both.

-s *groupname pathname*

changes groups only for the one directory or file specified by its pathname (relative to your home storage directory). Using **-s** disables all other CHGRPSTG options except **-v**. Note the syntax difference from CHMODSTG: here, the group name precedes the pathname immediately after **-s**.

-R recursively includes all the children (subdirectories and stored files) of the directory specified on the execute line. You can combine **-R** with other options (except **-s**) to further control CHGRPSTG's scope of action.

Interaction options:

-i prompts for your yes/no confirmation for every directory or stored file that CHGRPSTG tries to change. Any response except YES is treated as NO; you can NOT supply different groups for different files by using **-i**.

-v interactively reports the group change made for every directory or stored file that CHGRPSTG changes (e.g., "changed from oldgrp to newgrp"). You can combine **-v** with CHGRPSTG's **-i** prompting option, or use it for confirmation reports even without prompts.

-h displays the CHGRPSTG help package (a brief list of options). Help cannot be combined with any other options.

HTAR (Manage Stored File Collections)

ROLE.

On LC production machines (but not at other ASC sites), HTAR is a separate, locally developed utility program that serves as a special-purpose front end to the parallel FTP daemons for storage access. HTAR combines a flexible file bundling tool (like TAR) with fast parallel access (it acts as an alternative to the PFTP client) to open and secure STORAGE, to let you store and selectively retrieve even very large sets of files very efficiently. (Invoking HTAR's -F option lets you generalize these features for fast, file-bundled transfer to *non*-STORAGE locations as well.)

FEATURES.

HTAR's enhanced features include:

- Imposes no limit on the total size of the archives that it builds (some have successfully reached 200 Gbyte and 1,000,000 member files) and accepts input files (archive members) as large as 68 Gbyte.
- Uses a TAR-like syntax and supports TAR-compatible archive files by relying on the POSIX 1003.1 TAR file format.
- Bundles files in memory using multiple concurrent threads and transfers them into an archive file built *directly* in storage by default, to avoid needing extra local online disk space.
- Takes advantage of available parallel interfaces to storage to provide fast file transfers (measured at as high as 150 Mbyte/s, over 30 times the typical rate for transferring small files separately).
- Uses an external index file to easily accommodate thousands of small files in any archive, and to support retrieval of specified files from within a still-stored archive without first retrieving the whole archive from HPSS. (WARNING: you can use filters such as * to *create* an HTAR archive but you CANNOT reliably use filters to *retrieve* files from within an already stored HTAR archive. See the "Retrieving Files" section of the HTAR Reference Manual (URL: <http://www.llnl.gov/LCdocs/htar/index.jsp?show=s4.2>) for possible workarounds.)
- Allows easily building and storing incremental archives (consisting of only recently changed files).

EXECUTE LINES.

When the storage system (HPSS) is up and available to users you can execute HTAR with a command line that has the general form

```
htar action archive [options] [filelist]
```

and the specific form

```
htar -c|t|x|X|K -f archive [-BdEFhHILmMoOpSTvVwY] [flist]
```

where exactly one *action* and the *archive* are always required, while the control options and (except when using -c) the *filelist* can be omitted (and the options can share a hyphen flag with the action for convenience). Users familiar with TAR can guess how to run HTAR from this model (although there

are some tricky syntax differences). Others should consult the [HTAR Reference Manual](http://www.llnl.gov/LCdocs/htar) (URL: <http://www.llnl.gov/LCdocs/htar>) for usage suggestions, annotated examples, technical tips, full option details, and known problems.

One unusual feature of HTAR lets you not only avoid retrieving an entire archive from storage before extracting specified member files from within it, but also lets you (optionally) avoid even staging tape-resident archive files to HPSS disk before extracting specified members directly to your local machine. The NOSTAGE suboption of HTAR's -H (uppercase) control option lets you quickly retrieve (small) files from within a (much larger) stored-on-tape archive file, while leaving the archive on tape. For example, to retrieve file TEST5 from within the archive MYPROJ.TAR, stored in the PROJECTS subdirectory of your HPSS home directory, while leaving the whole archive still stored on tape, you could use

```
htar -x -f projects/myproj.tar -H nostage test5
```

HOPPER FRONT END.

You can use LC's visual file-transfer tool (controller), called HOPPER, to run HTAR as a controllee if you wish (it allows CTRL-CLICK file selection). But to do so you must copy all of your target files to HOPPER's CLIPBOARD, since "Make HTAR Archive" is only available as an operation (submenu choice) on clipboard entries. See the HOPPER section of the [HTAR manual](http://www.llnl.gov/LCdocs/htar/index.jsp?show=s2.5) (URL: <http://www.llnl.gov/LCdocs/htar/index.jsp?show=s2.5>) for the details. To list the contents of an existing HTAR archive (same operation as -t with HTAR), find the archive in a HOPPER directory table and *double* click on its icon (note that unlike HTAR itself, HOPPER never includes the archive's checksum file in its contents report). To extract from an existing HTAR archive, copy the desired files to the CLIPBOARD.

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Keyword Index

To see an alphabetical list of keywords for this document, consult the [next section](#) (page 40).

Keyword	Description
<u>entire</u>	This entire document.
<u>title</u>	The name of this document.
<u>scope</u>	Topics covered in EZSTORAGE.
<u>availability</u>	Where these programs run.
<u>who</u>	Who to contact for assistance.
<u>introduction</u>	Role and goals of EZSTORAGE.
<u>overview</u>	LC storage strengths and weaknesses.
<u>storage-summary</u>	Storage system constraints and common commands.
<u>storage-interfaces</u>	Descriptions of STORAGE access pathways.
<u>ftp-overview</u>	Brief description of FTP interface.
<u>nft-overview</u>	Brief description of NFT interface.
<u>additional-interfaces</u>	Alternative interface options.
<u>accessing-storage</u>	Offsite access strategies compared.
<u>storage-copies</u>	Multiple copies of same stored file.
<u>ftp</u>	Using FTP to transfer files.
<u>file-transfer-protocol</u>	Using FTP to transfer files.
<u>ftp-commands</u>	Basic FTP options explained.
<u>ftp-example</u>	Sample file transfer with FTP.
<u>ftp-pitfalls</u>	Wildcard dangers with stored files.
<u>hsi</u>	Using the Hierarchical Storage Interface.
<u>using-hsi</u>	Tool to list stored files.
<u>how-to-run-hsi</u>	How to run HSI.
<u>hsi-ftp-compatibility</u>	HSI FTP Compatibility
<u>hsi-examples</u>	HSI Examples
<u>nft</u>	Using NFT to transfer files.
<u>network-file-transfer</u>	Using NFT to transfer files.
<u>nft-syntax</u>	Specifying sec. levs, hosts with NFT.
<u>nft-commands</u>	Basic NFT options by task.
<u>nft-example</u>	Sample file transfer with NFT.
<u>sharing-files</u>	Sharing stored files.
<u>file-sharing</u>	Sharing stored files.
<u>storage-groups</u>	Using storage groups.
<u>permissions</u>	Setting permissions by storage group.
<u>reading-shared-files</u>	Sharing by the reader.
<u>storage-tools</u>	Three LC storage-helper tools.
<u>lstorage</u>	Tool to list stored files.
<u>chmodstg</u>	Tool to change storage permissions.
<u>chgrpstg</u>	Tool to change storage groups.
<u>htar</u>	Tool to bundle files into storage archives.
<u>index</u>	The structural index of keywords.
<u>a</u>	The alphabetical index of keywords.
<u>date</u>	The latest changes to this document.
<u>revisions</u>	The complete revision history.

Alphabetical List of Keywords

Keyword	Description
<u>a</u>	The alphabetical index of keywords.
<u>accessing-storage</u>	Offsite access strategies compared.
<u>additional-interfaces</u>	Alternative interface options.
<u>availability</u>	Where these programs run.
<u>chgrpstg</u>	Tool to change storage groups.
<u>chmodstg</u>	Tool to change storage permissions.
<u>date</u>	The latest changes to EZSTORAGE.
<u>entire</u>	This entire document.
<u>file-sharing</u>	Sharing stored files.
<u>file-transfer-protocol</u>	Using FTP to transfer files.
<u>ftp</u>	Using FTP to transfer files.
<u>ftp-commands</u>	Basic FTP options explained.
<u>ftp-example</u>	Sample file transfer with FTP.
<u>ftp-overview</u>	Brief description of FTP interface.
<u>ftp-pitfalls</u>	Wildcard dangers with stored files.
<u>how-to-run-hsi</u>	How to run HSI.
<u>hsi</u>	Using the Hierarchical Storage Interface.
<u>hsi-examples</u>	HSI Examples
<u>hsi-ftp-compatibility</u>	HSI FTP Compatibility
<u>htar</u>	Tool to bundle files into storage archives.
<u>index</u>	The structural index of keywords.
<u>introduction</u>	Role and goals of EZSTORAGE.
<u>lstorage</u>	Tool to list stored files.
<u>network-file-transfer</u>	Using NFT to transfer files.
<u>nft</u>	Using NFT to transfer files.
<u>nft-commands</u>	Basic NFT options by task.
<u>nft-example</u>	Sample file transfer with NFT.
<u>nft-overview</u>	Brief description of NFT interface.
<u>nft-syntax</u>	Specifying sec. levs, hosts with NFT.
<u>overview</u>	LC storage strengths and weaknesses.
<u>permissions</u>	Setting permissions by storage group.
<u>reading-shared-files</u>	Sharing by the reader.
<u>revisions</u>	The complete revision history.
<u>scope</u>	Topics covered in EZSTORAGE.
<u>sharing-files</u>	Sharing stored files.
<u>storage-copies</u>	Multiple copies of same stored file.
<u>storage-groups</u>	Using storage groups.
<u>storage-interfaces</u>	Descriptions of STORAGE access pathways.
<u>storage-summary</u>	Storage system constraints and common commands.
<u>storage-tools</u>	Three LC storage-helper tools.
<u>title</u>	The name of this document.
<u>using-hsi</u>	Tool to list stored files.
<u>who</u>	Who to contact for assistance.

Date and Revisions

Revision Date -----	Keyword Affected -----	Description of Change -----
24Sep09	<u>introduction</u> <u>ftp-commands</u> <u>storage-groups</u> <u>chgrpstg</u>	removed NETMON references. removed NETMON references. removed LDAP references. removed LDAP references.
01Mar09	<u>hsi</u> <u>using-hsi</u> <u>how-to-run-hsi</u> <u>hsi-ftp-compatibility</u> <u>hsi-examples</u> <u>macintosh-problems</u> <u>file-format</u> <u>file-name-problems</u> <u>suntar</u>	Using the Hierarchical Storage Interface. Tool to list stored files. How to run HSI. HSI FTP Compatibility HSI Examples Deprecated and removed from document. Deprecated and removed from document. Deprecated and removed from document Deprecated and removed from document.
12Sep07	<u>storage-summary</u> <u>storage-tools</u> <u>htar</u>	HTAR member limit now 68 Gbyte. Obsolete machines deleted. HTAR member limit now 68 Gbyte.
23Apr07	<u>storage-summary</u> <u>nft-overview</u> <u>storage-copies</u> <u>ftp-example</u> <u>nft-syntax</u> <u>nft-commands</u>	NFT SETCOS command added. NFT storage "routing" noted. NFT SETCOS command added. Updated, ATLAS replaces GPS. NFT storage "routing" noted. NFT SETCOS command added.
01Nov06	<u>sharing-files</u> <u>storage-groups</u> <u>chgrpstg</u>	Improved subdivision for clarity. LDAPSEARCH replaces DCECP tool, pitfalls and execution details revealed. Cross ref to LDAPSEARCH added.
12Jul06	<u>ftp-overview</u> <u>ftp-commands</u> <u>ftp-pitfalls</u> <u>storage-groups</u>	Warning about different FTP clients. Pitfalls cross ref added. Warning about different FTP clients. Few machines now support DCECP.
10Apr06	<u>storage-summary</u> <u>nft-overview</u>	Changed storage homes for LANL, Sandia users. LANL, Sandia users can now run NFT.
12Dec05	<u>htar</u> <u>additional-interfaces</u>	Limitations with HOPPER noted. Cross ref to HOPPER details added.
13Jul05	<u>accessing-storage</u> <u>htar</u>	Time limit dropped to 2 hr. New features added.

additional-interfaces
HOPPER's storage role summarized.

06Dec04 accessing-storage
12-hour timeout noted; cross ref.

19Oct04 storage-summary
NFT ACL commands deleted.
nft-overview NFT ACL commands discontinued.
nft-syntax NFT ACL commands discontinued.
nft-commands ACL commands deleted from table.

15Sep04 introduction HTAR role clarified.
overview HTAR parallel transfers clarified.
htar Options and limits updated.

07Jun04 nft-syntax HPSS error message format noted.
a Sorting error fixed in index.

17Feb04 storage-summary
COS and ACL commands added.
ftp-pitfalls Recursive NFT deletes now enabled.
nft-syntax Five new ACL commands noted.
nft-commands Five ACL commands added to table.
nft-example Dialog updated, details added.

18Nov03 htar Speed and size details updated.
storage-summary Maximum file size updated.

11Aug03 storage-summary
Cross ref. to HTAR added.
ftp-commands Cross ref. to HTAR added.
nft-syntax Cross ref. to HTAR added.
htar NOSTAGE feature explained, illustrated.

12May03 introduction NETMON units user controlled now.
ftp-commands PARALLEL has altered role.
storage-tools Now under Linux/CHAOS too.

19Feb03 introduction SFTP is not a storage interface.
overview SFTP is not a storage interface.
additional-interfaces SFTP is not a storage interface.

21Nov02 introduction FTP monitored by class of service now.
ftp-commands FTP monitored by class of service now.
storage-copies Another role for COS noted.

08Oct02 storage-tools Forest departs, availability clarified.

17Jun02 ftp-pitfalls New section with MDELETE warning.
ftp-commands DELETE, MDELETE added.
nft-commands DELETE added.
storage-copies New section on duplicate copies.
storage-summary Maximum file size FTP vs. HTAR.
index New keywords for new sections.

02May02	<u>htar</u>	Warning added on HTAR retrievals.
05Feb02	<u>introduction</u> <u>ftp-commands</u> <u>ftp-example</u> <u>storage-tools</u> <u>storage-groups</u>	NETMON cross reference added. NETMON cross reference added. Example dialog updated. Not available under Linux. WEST reference replaced.
27Aug01	<u>overview</u> <u>scope</u> <u>storage-tools</u> <u>htar</u> <u>index</u>	HTAR role added. HTAR availability noted. HTAR role added. New summary section added. New keyword for new section.
09Jul01	<u>overview</u> <u>storage-interfaces</u> <u>ftp-commands</u> <u>ftp-example</u>	PFTP role added. Automatic parallel FTP to storage. PFTP for storing files elsewhere. PARALLEL local command added. Automatic parallel transfers shown.
17Apr01	<u>overview</u> <u>accessing-storage</u> macintosh-problems	VPN role noted for offsite FTP. Three strategies revised re VPN, IPA. VPN role noted for offsite FTP.
26Mar01	<u>ftp-overview</u> <u>ftp-commands</u> <u>ftp-example</u> <u>nft-syntax</u>	Parallel client now the default. Parallel client now the default. Verbose, preauthenticated parallel dialog. No verbosity change for NFT.
25Sep00	<u>sharing-files</u>	Link to file-sharing comparison added.
01Aug00	<u>storage-tools</u>	Now on all LC production machines.
22May00	<u>storage-tools</u> <u>introduction</u> <u>sharing-files</u> <u>index</u>	LSTORAGE, CHMODSTG, CHGRPSTG explained. New sections cross referenced. Relevant new tools cited. New keywords for new sections.
11Apr00	<u>availability</u> <u>overview</u> <u>accessing-storage</u>	Revised print-file instructions. FTP gateway discontinued. FTP gateway discontinued.
10Jan00	entire	Revised first edition of LC EZSTORAGE.
13Dec99	entire	Draft edition of LC EZSTORAGE manual.

ANG (24Sep09)

UCRL-WEB-200719

Privacy and Legal Notice (URL: <https://lc.llnl.gov/disclaimer.html>)

ANG (24Sep09) Contact: lc-hotline@llnl.gov