Using Shift for Local and Remote Transfers (Recommended)

The NAS-developed Shift tool can copy files locally on NAS enclave hosts, transfer files between hosts inside the NAS enclave, and transfer files between the NAS enclave and remote hosts. You can also use Shift to check the status of transfers at any time, receive email notification of completion, errors, and warnings, and restart interrupted transfers or transfers with errors.

All functionality is accessed through the Shift client, which is invoked via the `shiftc` command. The syntax for `shiftc` is similar to the syntax for the `cp` and `scp` commands.

Shift is the recommended method for transferring files to and from the Lou mass storage system, as it can create tar files as part of the transfer and split a transfer into multiple tar files for oversized directories (larger than 1 TB).

Advanced Features

Shift includes the following advanced features:

- Automatic parallelization of transfers
- Local and remote tar creation and extraction
- Synchronization based on modification times and checksums (similar to `rsync`)
- Automatic file integrity verification and correction
- Ability to restart transfers
- Automatic retrieval of files from tape storage (DMF-managed Lou filesystems)
- Ability to check status of current transfers

How to Use Shift

See the following articles for detailed information about how to use Shift:

- Using Shift for Local Transfers and Tar Operations
- Using Shift for Transfers and Tar Operations Between Two NAS Hosts
- Using Shift for Remote Transfers and Tar Operations
- Checking Shift Transfer Status and Restarting Transfers
- Shift Command Options

Additional Resources

You can also see presentation slides for three HECC training webinars that demonstrate how to use the tool:

- Simplifying and Optimizing Your Data Transfers (PDF)
- Advanced Features of the Shift Automated File Transfer Tool (PDF)
- Simple Automated File Transfers Using SUP and Shift (PDF)

Recordings of each presentation are also available in the Past Webinars Archive.

Note: Some hostnames and options may have changed since the webinars were presented.

Shift can provide status on transfers currently in progress, and can report results of transfers that were performed within the past week. You can restart and complete any interrupted transfers or transfers that completed with errors.

Note: Transfers that have been inactive for more than a week cannot be restarted.

- Check transfer status from a NAS high-end computing host (for example, pfe*21*):

  pfe*21*% shiftc --status

- Check transfer status from remote host (for example, your local system):

  your_local_system% sup shiftc --status

- Restart a transfer that has errors or was interrupted:

  pfe*21*% shiftc --restart --id=*transfer_id*

Checking Transfer Status

Show the status of all transfers:

```
pfe21% shiftc --status
id | state | dirs | files |     file size |  date |    run |    rate
   |       | sums | attrs |      sum size |  time |   left |
---+-------+------+-------+---------------+-------+--------+---------
 1 | done  |  0/0 |   1/1 |     92KB/92KB | 10/03 |     2s |   46KB/s
   |       |  0/0 |   0/0 |     0.0B/0.0B | 17:06 |        |
 2 | done  |  0/0 |   1/1 |     92KB/92KB | 10/03 |     8s | 11.5KB/s
   |       |  0/0 |   1/1 |     0.0B/0.0B | 17:06 |        |
 3 | done  |  1/1 |   2/2 |     99KB/99KB | 10/03 |     1s |   99KB/s
   |       |  4/4 |   0/0 |   198KB/198KB | 17:07 |        |
 4 | error |  1/1 |   1/2 |     92KB/99KB | 10/03 |     3s | 30.7KB/s
   |       |  0/0 |   0/0 |     0.0B/0.0B | 17:08 |        |
 5 | done  |  1/1 | 64/64 | 65.5GB/65.5GB | 10/03 |    29s | 2.26GB/s
   |       |  0/0 |   0/0 |     0.0B/0.0B | 17:09 |        |
```

Showing Detailed Status

The examples in this section show how to display details about specific transfers that are listed in the output in the previous example.

TIP: Showing the detailed status of a single transfer might result in extremely large output, as a single transfer may contain hundreds, thousands, or even millions of transferred files—and multiple lines may be output for each file. Therefore, if you want to see the detailed status of a very large transfer, redirect the output to a file. For example:

```
shiftc --status --id=5 > status.id5
```

You can also search for files matching a given pattern, and show only their status. For example:

```
shiftc --status --id=1 --search=file2
```

Show the detailed status of all operations in transfer #2, from your local system:

```
your_local_system% sup shiftc --status --id=2
state | op     | target              | size |  date |    run |   rate
      | tool   | info                |      |  time |   left |
------+--------+---------------------+------+-------+--------+-------
done  | cp     | lfe2:/u/user1/file1 | 92KB | 10/03 |     5s | 18KB/s
      | rsync  | -                   |      | 17:06 |        |
done  | chattr | lfe2:/u/user1/file1 |    - | 10/03 |     1s |      -
      | sftp   | -                   |      | 17:06 |        |
```

Show the detailed status of all operations in transfer #3 that involve a filename containing *file2*:

```
pfe21% shiftc --status --id=3 --search=file2
state | op    | target              | size |  date |    run |  rate
      | tool  | info                |      |  time |   left |
------+-------+---------------------+------+-------+--------+------
done  | cp    | /username/dir2/file2 |  7KB | 10/03 |     1s | 7KB/s
      | mcp   | -                   |      | 17:07 |        |
done  | cksum | /username/dir2/file2 |  7KB | 10/03 |     1s | 7KB/s
      | msum  | -                   |      | 17:07 |        |
```

Show the detailed status of all operations in transfer #4 that have an error state:

```
pfe21% shiftc --status --id=4 --state=error
```

```
state | op    | target                    | size |  date |    run | rate
      | tool  | info                      |      |  time |   left |
------+-------+---------------------------+------+-------+--------+-----
error | cp    | /username/dir2/file2      |  7KB |     - |      - |    -
      | rsync | rsync: send_files         |      |       |        |
      |       | failed to open            |      |       |        |
      |       | "/username/dir2/file2":   |      |       |        |
      |       | Permission denied         |      |       |        |
```

Showing Transfer History

Show the history of all transfers:

```
pfe21% shiftc --history
id | origin             | command
---+--------------------+-------------------------------------------------------------------------
 1 | pfe21.nas.nasa.gov | shiftc file1 /u/username/dir1
   | [/u/user1]         |
 2 | pfe21.nas.nasa.gov | shiftc -p file1 lfe2:
   | [/u/user1]         |
 3 | your_local_system  | sup shiftc -r --no-verify /username/dir1 lfe2:/username/dir2
   | [/Users/user1]     |
 4 | your_local_system  | sup shiftc -r --secure lfe2:/username/dir2 .
   | [/username]        |
 5 | pfe21.nas.nasa.gov | shiftc -r --hosts=2 /nobackup/user1/bigdir1 /nobackup/user1/bigdir2
   | [/u/user1]         |
```

Show the history of all transfers that involve a host or a command containing lfe2 from your local system:

```
your_local_system% sup shiftc --history --search=lfe2
id | origin             | command
---+--------------------+---------------------------------------
 2 | pfe21.nas.nasa.gov | shiftc -p file1 lfe2:
   | [/u/user1]         |
 4 | your_local_system  | shiftc -r --secure lfe2:/username/dir2 .
   | [/username]        |
```

Options for the shiftc command are described in this section. For more information, including syntax and examples, see **man shiftc** on a PFE or LFE.

Initialization Options

Transfers are initialized using syntax identical to cp and scp commands for local and remote transfers, respectively.

--clients=NUM
: Parallelize the transfer by using additional clients on each host. If the number given is 1, no additional clients will be used. A number greater than 1 will fork additional processes on each host to more fully utilize system resources and improve transfer performance.

--create-tar
: Create a tar file of all sources at the destination, which must be a non-existing filename. This option implies --recursive and --no-offline. By default, multiple tar files are created at 1 TB boundaries. The split size may be changed or splitting disabled using the --split-tar option. The --index-tar option may be used to produce a table of contents file for each tar file created. Note that this option cannot be used with --sync.

Create any missing parent directories. This option allows files to be transferred to a directory hierarchy that may not already exist, similar to the -d option of the install command.

-L, --dereference
: Always follow symbolic links to both files and directories. Note that this can result in file and directory duplication at the destination as all symbolic links will become real files and directories.

-d, --directory
: Create any missing parent directories. This option allows files to be transferred to a directory hierarchy that may not already exist, similar to the -d option of the install command.

--exclude=REGEX
: Do not transfer source files matching the given regular expression. Note that regular expressions must be given in Perl syntax (for details, see perlre(1) on The Perl Foundation website) and should be quoted on the command line when including characters normally expanded by the shell (such as "*"). Shell wildcard behavior can be approximated by using ".*" in place of "*".

--extract-tar
: Extract all source tar files to the destination, which must be an existing directory or non-existing directory name. This option implies --no-offline. Note that only tar archives in the POSIX ustar format are supported, but GNU extensions for large UIDs, GIDs, file sizes, and filenames are handled appropriately. Also note that this option cannot be used with --sync.

-f, --force
: Overwrite existing read-only files at the destination by temporarily adding owner write permission. File permissions will be restored later in the transfer. Note, however, that if the transfer does not complete successfully, files may be left with the wrong permissions. Also note that files marked as immutable using chattr +i cannot be overwritten even when this option is in effect.

--host-file=FILE
: Parallelize the transfer by using additional clients on the hosts specified in the given file (one hostname per line). This option implies a value for the --hosts option that is equal to the number of hosts in the file plus any additional hosts from the --host-list option. Fewer hosts may be used by explicitly specifying a value for the --hosts option. Note that the actual number of client hosts used will depend on the number of hosts that have equivalent access to the source and/or destination filesystems. Within PBS job scripts, this option can be set to the $PBS_NODEFILE variable to use all nodes of the job.

--host-list=LIST
: Parallelize the transfer by using additional clients on the hosts specified in the given comma-separated list. This option implies a value for the --hosts option that is equal to the number of hosts on the list plus any additional hosts from the --host-file option. Fewer hosts may be used by explicitly specifying a value for the --hosts option. Note that the actual number of client hosts used will depend on the number of hosts that have equivalent access to the source and/or destination filesystems.

--hosts=NUM
: Parallelize the transfer by using additional clients on (at most) the given number of hosts. If the number given is 1, no additional clients will be used. A number greater than 1 enables automatic transfer parallelization where additional clients may be invoked on additional hosts to increase transfer performance. Note that the actual number of clients used will depend on the number of hosts for which Shift has filesystem information and the number of hosts that have equivalent access to the source and/or destination filesystems. Client hosts will be accessed as the current user with host-based authentication or an existing ssh agent that contains an ssh from a file matching ~/.ssh/id*.

--identity=FILE
: Authenticate to remote systems using the given ssh identity file. The corresponding public key must reside in the appropriate user's ~/.ssh/authorized_keys file on the remote host. Note that only identity files without passphrases are supported. If a passphrase is required, an ssh agent may be used instead, but with a loss of reliability. This option is not needed if the remote host accepts host-based authentication from client hosts.

-I, --ignore-times
: By default, the --sync option skips the processing of files that have the same size and modification time at the source and destination. This option specifies that files should always be processed by checksum regardless of size and modification time.

--include=REGEX
: Only transfer source files matching the given regular expression. Note that regular expressions must be given in Perl syntax (see perlre(1) on the Perl Foundation website for details) and should be quoted on the command line when including characters normally expanded by the shell (such as "*"). Shell wildcard behavior can be approximated by using ".*" in place of "*".

--index-tar

Create a table-of-contents file for each tar file created with the --create-tar option. The table of contents will show each file in the tar file along with permissions, user/group ownership, and size. For a tar file "file.tar", the table of contents will be named "file.tar.toc". Unless the --no-verify option is used, a checksum file named "file.tar.sum" will also be created, which is suitable as input for msum --check-tree -c. Note that when the --split-tar option is used, multiple table-of-contents files may be created. For each split tar file "file.tar-i.tar", the table of contents will be named "file.tar--.tar.toc" and the checksum file will be named "file.tar-i.tar.sum".

--newer=[TYPE:]DATE

Only transfer source files whose modification time (or combination of modification, access, and/or creation times) is newer (inclusive) than the given date. Any date string supported by the Perl Date::Parse module (see Date::Parse(3) for details) can be specified. An optional type expression of the form "[acmACM]+(|[acmACM]+)*)" can be given to specify conditions in which one or more conditions are or are not newer than the date, where: "a" is access time; "c" is creation time; "m" is modification time; and "A", "C", and "M" are their inverses, respectively. For example, "aM|cm" would transfer source files whose access time was newer than the date but whose modification time was not newer, or files whose creation time and modification time were newer. Note that this option can be combined with --older to specify exact date ranges.

-P, --no-dereference

Never follow symbolic links to files or directories. Note that this can result in broken links at the destination, as files and directories referenced by symbolic links that were not explicitly transferred or implicitly transferred using --recursive might not exist on the target.

-T, --no-target-directory

Do not treat the destination specially when it is a directory or a symbolic link to a directory. This option can be used with recursive transfers to copy a directory's contents into an existing directory instead of into a new subdirectory beneath it as is done by default.

--older=[TYPE:]DATE

Only transfer source files whose modification time (or combination of modification, access, and/or creation times) is older than the given date. Any date string supported by the Perl Date::Parse module (see Date::Parse(3) for details) can be specified. An optional type expression of the form "[acmACM]+(|[acmACM]+)*)" can be given to specify conditions in which one or more conditions are or are not older than the date, where: "a" is access time; "c" is creation time; "m" is modification time; and "A", "C", and "M" are their inverses, respectively. For example, "aM|cm" would transfer source files whose access time was older than the date but whose modification time was not older, or files whose creation time and modification time were both older. Note that this option can be combined with --newer to specify exact date ranges.

--pipeline

Produce verified files earlier in the transfer by preferring to process the normal sequence of operations (find, copy, checksum, verify checksum, change attributes) in reverse order. In default (non-pipeline) operation, these stages are performed in order where all files are found before any are copied; before any are checksummed, etc. When this option is enabled, files that have reached the change attribute stage will be processed before files that have reached the verify checksum stage, which will be processed before files that have reached the checksum stage, etc. This allows you to perform parallel processing on verified files while the transfer is still ongoing. To determine the list of files that have been successfully verified in a transfer with id "N", use --status --id=N --state=done --search=chattr. When multiple clients are participating in the transfer (i.e., --clients or --hosts options are specified with a value greater than 1), different clients will prefer different stages for more overlap of reads and writes between the source and destination filesystems.

Note that while several strategies are employed to ensure that checksums are computed from disk and not from cache, it is safest to use this option only when there is actually a need to process destination files during the transfer.

--ports=NUM1:NUM2

Use ports from the range NUM1-NUM2 for the data streams of TCP-based transports (currently bbftp and fish-tcp). All connections originate from the client host so the given port range must be allowed on the network path to the remote host and by the remote host itself.

-R, -r, --recursive

Transfer directories recursively. This option implies --no-dereference. Note that any symbolic links pointing to directories that are given on the command line will be followed during recursive transfers (identical to the default behavior of the cp command).

--secure

Encrypt data during remote transfers and use secure ciphers and MACs with SSH-based transports. Note that this option will, in most cases, decrease performance as it eliminates some higher performance transports and increases CPU utilization during SSH connections.

--sync

Synchronize files between the source and destination, similar to the rsync command. By default, files that have the same size and modification time at the source and destination will not be transferred. If the size or modification time of a file differs between the two, the contents of the file will be compared via checksum and any portions that differ will be transferred to the destination. To skip the size and modification time checks and always begin with the checksum stage, use -I or --ignore-times. If the --no-verify option is specified, integrity verification is not performed; this will increase performance when there are many files at the source that are not at the destination, but will decrease performance when there are large files that have only small changes between the source and destination. Setting the --retry option to zero with this option can be used to show which files differ without making any changes. Note that when syncing directories, the destination should be specified as the parent of the location where the source directory should be transferred to. Also note that this option cannot be used with the --create-tar or --extract-tar options.

--user=USER

Set the user that will be used to access remote systems.

--wait

Block until the transfer completes and print a summary of the transfer. This option implies --no-mail. An exit value of 0 indicates that the transfer has successfully completed while an exit value of 1 indicates that the transfer has failed or that the waiting process was terminated prematurely. This option may be used together with --monitor to show the real-time status of the transfer while waiting.

Feature Disablement Options

The following options disable certain default features.

--no-cron
: Do not attempt to recover from host/process failures via cron. Note that when such a failure occurs, the transfer will become stuck in the "run" state until stopped.

--no-mail[=LIST]
: By default, emails are sent when a transfer completes successfully, aborts with errors, or is stopped; and for the first instances of alerts, errors, throttling, and/or warnings while running. This option prevents emails from being sent altogether or, optionally, for a specific subset of states. The given list may be a comma-separated subset of {alert, done, error, run, stop, throttle, warn}. This option may be desirable when performing a large number of scripted transfers. Note that equivalent transfer status and history information can always be manually retrieved using the --status and --history options, respectively.

--no-offline
: By default, files transferred to and from DMF-managed filesystems will be migrated to offline media as soon as the transfer is completed. This option specifies that files should instead be kept online (not migrated). Note that DMF may still choose to migrate a file even when this option is enabled.

--no-preserve[=LIST]
: By default, times, permissions, ownership, striping, ACLs, and extended attributes of transferred files and directories are preserved when possible. This option specifies that these items (or an optional specified subset) should not be preserved. The given list may be a comma-separated subset of {acl, mode, owner, stripe, time, xattr}. Note that permissions may be left in various states depending on the invoking user's umask and the transport utilized. In particular, read access at the destination may be more permissive than read access at the source.

--no-recall
: By default, files transferred from DMF-managed filesystems will be recalled from offline media as soon as the transfer begins and again before each batch of files is processed. This option specifies that files should not be recalled. Note that DMF will still recall files as needed even when this option is enabled.

--no-sanity
: Disable file existence and size checks at the end of the transfer. This option was included for benchmarking and completeness purposes and is not recommended for general use.

--no-silent
: By default, the checksums of all files transferred with Shift are stored in a per-user database. When a file with a known checksum is transferred and has not been modified since the checksum was stored, the transfer will be put into the "alert" state if the current checksum does not match the stored checksum. This option disables the storage of checksums and comparison against existing checksums. While silent corruption detection adds minimal overhead during normal operation, it can increase the probability of lock contention when there are large numbers of clients.

--no-verify
: By default, files are checksummed at the source and destination to verify that they have not been corrupted and if corruption is detected, the corrupted portion of the destination file is automatically corrected using a partial transfer from the original source. This functionality decreases the performance of transfers in proportion to the file size. If assurance of integrity is not required, the --no-verify option may be used to disable verification.

History, Management, and Status Options

Once one or more transfers have been initialized, you may view transfer history, stop/restart transfers, and/or check transfer status with the following options.

--history[=csv]
: Show a brief history of all transfers including the transfer identifier, the origin host/directory, and the original command. When --history=csv is specified, history is shown in CSV format.

--id=NUM
: Specify the transfer identifier to be used with management and status commands.

--last-sum
: Query the silent corruption database for all files given on the command line and print (one file per line) the last known checksum, the file modification time associated with this checksum, and the filename. When --index-tar is given, the first file argument is assumed to be a tar file and the remaining arguments names of files within the tar for which checksum information will be printed. A checksum of "-" means that no information is stored for the file.

--mgr=HOST
: Set the host that will be used to manage transfers. By default, this host will be accessed as the current user with host-based authentication or an existing ssh agent. The user and/or identity used to access the manager host may be changed with the--mgr-user and --mgr-identity options, respectively.

--mgr-identity=FILE
: Authenticate to the manager host using the given ssh identity file. The corresponding public key must reside in the appropriate user's ~/.ssh/authorized_keys file on the manager host. Note that only identity files without passphrases are supported. If a passphrase is required, an ssh agent may be used instead, but with a loss of reliability. This option is not needed if the manager host accepts host-based authentication from client hosts.

--mgr-user=USER
: Set the user that will be used to access the manager host. Note that if the transfer is initiated by root and the-mgr-identity option is not specified, manager communication will be performed as the given user, so that user must be authorized to run processes locally. In particular, care should be taken on PBS-controlled nodes, where the given user should either own the node or be on the user exception list.

--monitor[=FORMAT]
: Show the realtime status of all running transfers including the transfer identifier, the current state, the number of directories completed, the number of files transferred, the number of files checksummed, the number of attributes preserved, the amount of data transferred, the amount of data checksummed, the time the transfer started, the duration of the transfer, the estimated time remaining in the transfer, and the rate of the transfer. Note that updates are realtime with respect to the information available to the manager and not with respect to the transports that may be carrying out the transfer. Status will

be returned in CSV format when the --monitor=csv is specified. Duration and estimated time will be zero-padded when--monitor=pad is specified. When --monitor=color is specified, transfers in the {error, run, throttle, warn} states will be shown with {red, green, magenta, yellow} coloring, respectively. When --id is specified, only the given transfer will be shown. When all transfers (or the one specified) have completed, the command will exit. This option may be used with the --wait option to monitor progress while waiting.

--plot=[=[BY[:/]]LIST]

Produce output suitable for piping into gnuplot (version 5 or above) that shows detailed performance over time across all transfers. The --id and --state options may be used to plot only a single transfer or transfers in a particular state, respectively. The default plot will show the aggregate performance of each I/O operation (i.e. cp, sum, and cksum) and the aggregate performance of each metadata operation (i.e. find, mkdir, ln, and chattr) across all of the user's transfers. Operations and/or additional groupings are shown on the left y-axis axis across time on the x-axis with heat-based coloring indicating MB/s for I/O operations or operations per second for metadata operations. In addition, aggregate I/O and metadata performance will be shown as an overlaid point plot with green and blue points, respectively.

The list of plotted items may be changed by giving a comma-separated list consisting of one or more of the stages {chattr, cksum, cp, find, io, ln, meta, mkdir, sum} and/or one or more of the tools {bbftp, fish, fish-tcp, mcp, msum, rsync, shift-cp, shift-sum}. Note that "io" is a shorthand for "cp,sum,cksum", "meta" is a shorthand for "find,mkdir,ln,chattr", and "tool" is a shorthand for "bbftp,fish,fish-tcp,mcp,msum,rsync,shift-cp,shift-sum".

The list of items may be grouped by any of {client, fs, host, id, net, user} by prefixing one of these terms to the list. For example, --plot=id:cp would show a plot of the copy performance achieved by each transfer id. When a grouping is given without a specific list of metrics (e.g., --plot=id), "io" is assumed. When a slash "/" is used instead of colon ":", a heatmap-based bubble plot will be created with the size of each circle indicating the relative size of the batch of operations. For example, --plot=fs/tool would show a plot of the performance that each tool achieved on each file system with relative batch size.

--restart[=ignore]

Restart the transfer associated with the given --id that was stopped due to unrecoverable errors or stopped explicitly via the--stop option. If --restart=ignore is specified, all existing errors will be ignored and the transfer will progress as if the associated files and directories were no longer part of the transfer. Note that transfers must be restarted on the original client host or one that has equivalent filesystem access. A subset of the available command-line options may be re-specified during a restart, including --bandwidth, --buffer, --clients, --cpu, --disk, --files, --force, --host-file, --host-list, --hosts, --interval, --io, --ior, --iow, --local, --net, --netr, --netw, --no-cron, --no-mail, --no-offline, --no-recall, --no-silent, --pipeline, --ports, --preallocate, --remote, --retry, --secure, --size, --streams, --stripe, --threads, and --window.

--search=REGEX

When the --status and --id options are specified, this option will show the full status of file operations in the associated transfer whose source or destination filename match the given regular expression. When the --history option is specified, this option will show a brief history of the transfers in which the origin host or original command matches the given regular expression. Note that regular expressions must be given in Perl syntax (see perlre(1) for details).

--state=STATE

When the ??status and ??id options are specified, --state=STATE will show the full status of file operations in the associated transfer that have the given state. When --id is not specified, this option will show the brief status of transfers in the given state. Valid states are done, error, none, queue, run, and warn. A state of "none" will show a summary of the given transfer.

--stats[=csv]

Show stats across all transfers including transfer counts, rates, tool usage, initialization options, error counts, and error messages. When --stats=csv is specified, stats are shown in CSV format without error messages.

--status[=FORMAT]

Show a brief status of all transfers including the transfer identifier, the current state, the number of directories completed, the number of files transferred, the number of files checksummed, the number of attributes preserved, the amount of data transferred, the amount of data checksummed, the time the transfer started, the duration of the transfer, the estimated time remaining in the transfer, and the rate of the transfer. When the number of transfers exceeds a set threshold (the default is 20), older successfully completed transfers beyond that limit will be omitted for readability. These omitted transfers can be shown using --status with --state=done. Status will be returned in CSV format when --status=csv is specified. Duration and estimated time will be zero-padded when --status=pad is specified. When --status=color is specified, transfers in the {done, error, run, stop, throttle, warn} states will be shown with {default, red, green, cyan, magenta, yellow} coloring, respectively.

When ??id is specified, --status[=FORMAT] will show the full status of every file operation in the associated transfer. For each operation, this includes the state, the type, the tool used for processing, the target path, associated information (error messages, checksums, byte ranges, and/or running host) when applicable, the size of the file, the time processing started, and the rate of the operation. Note that not all of these items will be applicable at all times (for example, "rate" will be empty if the state is "error"). Also note that operations are processed in batches so the rate shown for a single operation will depend on the other operations processed in the same batch. When --status=color is specified, operations in the {done, error, queue, run, warn} states will be shown with {default, red, cyan, green, yellow} coloring, respectively.

--stop

Stop the transfer associated with the given --id. Note that transfer operations currently in progress will run to completion but new operations will not be processed. Stopped transfers may be restarted with the --restart option.

Transfer Tuning Options

Some advanced options are available to tune various aspects of the shiftc command's behavior. These options are not needed by most users.

--bandwidth=BITS

Choose the TCP window size and number of TCP streams of TCP-based transports (currently, bbftp and fish-tcp) based on the given bits per second. The suffixes k, m, g, and t may be used for Kb, Mb, Gb, and Tb, respectively. The default bandwidth is estimated to be 10 Gb/s if a 10 GE adapter is found on the client host, 1 Gb/s if the client host can be resolved to an organization domain (by default, one of the six original generic top-level domains), and 100 Mb/s otherwise.

--buffer=SIZE

Use memory buffer(s) of the given size when configurable in the underlying transport being utilized (currently, all but rsync). The suffixes k, m, g, and t may be used for KiB, MiB, GiB, and TiB, respectively. The default buffer size is 4 MiB. Increasing the buffer size trades higher memory utilization for more efficient I/O.

--files=COUNT

Process transfers in batches of at least the given number of files. The suffixes k, m, b or g, and t may be used for 1E3, 1E6, 1E9, and 1E12, respectively. The default batch count is 1000 files. This option works in concert with --size and --interval to manage the number of checkpoints and the overhead of transfer management. A batch will initially consist of at least --files files or --size bytes, whichever is reached first. The batch may then be dynamically increased in size until there is enough work to span --interval seconds. To make batch selection completely dynamic, use --files=1 and --size=1.

--interval=SECS

Process transfers in batches that take about the given number of seconds. The default interval is 30 seconds. This option works in concert with --files and --size to manage the number of checkpoints, as well as the overhead of transfer management. A batch will initially consist of at least --files files or --size bytes, whichever is reached first. The batch may then be dynamically increased in size until there is enough work to span --interval seconds. Note that the actual time a batch takes will depend on its contents, and that the interval will be increased as the number of clients participating in a transfer increases to minimize contention for manager locks. To make batch selection completely static, use --interval=0.

--local=LIST

Specify one or more local transports to be used for the transfer in order of preference, separated by commas. Valid transports for this option currently include bbftp, cp, fish, fish-tcp, mcp, and rsync. Note that the given transport(s) will be given priority, but may not be used in some cases (for example, rsync is not capable of transferring a specific portion of a file as needed by verification mode). In such cases, the default transport based on File::Copy will be used. The tool actually used for each file operation can be shown using the --status option with the --id option set to the given transfer identifier.

--preallocate=NUM

Preallocate files when their sparsity is under the given percent, where sparsity is defined as the number of bytes a file takes up on disk divided by its size. Note that this option will only have an effect when the fallocate command is available, the destination file does not already exist, and the target filesystem properly supports fallocate's -n option. Also note that this option will not function properly when either bbftp or rsync (to a DMF filesystem) is utilized as the transport due to their use of temporary files.

--remote=LIST

Specify one or more remote transports to be used for the transfer in order of preference, separated by commas. Valid transports for this option currently include bbftp, fish, fish-tcp, rsync, and sftp. Note that the given transport(s) will be given priority, but may not be used in some cases (for example, bbftp is not capable of transferring files with spaces in their names and is also incompatible with the --secure option). In such cases, the default transport based on sftp will be used. The tool actually used for each file operation can be shown using --status with the --id option set to the given transfer identifier.

--retry=NUM

Retry operations deemed recoverable up to the given number of attempts per file. The default number of retries is 2. A value of zero disables retries. Note that disabling retries also disables the ability of --sync to change file contents. Also note that the given value is cumulative across all stages of a file's processing so different stages may not be retried the same number of times.

--size=SIZE

Process transfers in batches of at least the given total file size. The suffixes k, m, g, and t may be used for KB, MB, GB, and TB, respectively. The default batch size is 4 GB. This option works in concert with --files and --interval to manage the number of checkpoints and the overhead of transfer management. A batch will initially consist of at least --size bytes or --files files, whichever is reached first. The batch may then be dynamically increased in size until there is enough work to span --interval seconds. To make batch selection completely dynamic, use --files=1 and --size=1.

--split=SIZE

Parallelize the processing of single files using chunks of the given size. The suffixes k, m, g, and t may be used for KiB, MiB, GiB, and TiB, respectively. The default split size is zero, which disables single file parallelization. A split size of less than 1 GiB is not recommended. Lowering the split size will increase parallelism but decrease the performance of each file chunk and increase the overhead of transfer management. Raising the split size will have the opposite effect. The ideal split size for a given file is the size of the file divided by the number of concurrent clients available. Note that --split=SIZE does not have an effect unless the value of the --hosts option is greater than 1. Also note that --split=SIZE can, in some cases, decrease remote transfer performance as it eliminates some higher performance transports.

--split-tar=SIZE

Create tar files of around the given size when used with the --create-tar option. When multiple tar files are created for a destination tar file "file.tar", the resulting split tar files will be named "file.tar-i.tar" starting from "file.tar-1.tar". The suffixes k, m, g, and t may be used for KB, MB, GB, and TB, respectively. The default split tar size is 1 TB. A value of zero disables splitting. A split tar size of greater than 2 TB is not recommended. Note that resulting tar files may still be larger than specified when source files exist that are larger than the given size.

--streams=NUM

Use the given number of TCP streams in TCP-based transports (currently bbftp and fish-tcp). The default is the number of streams necessary to fully utilize the specified/estimated bandwidth using the maximum TCP window size. Note that it is usually preferable to specify the --bandwidth option, which allows an appropriate number of streams to be set automatically. Increasing the number of streams can increase performance when the maximum window size is set too low or there is cross-traffic on the network, but too high a value can decrease performance due to increased congestion and packet loss.

--stripe=[CEXP][::[SEXP][::PEXP]]

By default, a file transferred to a Lustre filesystem will be striped according to an administrator-defined policy (one stripe per GiB when not configured). It is recommended, although not required, that this policy preserve existing striping when the source resides on Lustre and has non-default striping. To disregard existing striping, "stripe" may be used with the --no-preserve=stripe option. To disable automatic striping completely and use the default Lustre behavior for all files and directories, use --stripe=0.

The user may override the default policy by specifying expressions for one or more of the stripe count (CEXP), stripe size (SEXP), and stripe pool (PEXP). For the stripe count, a positive number less than 65,536 indicates a fixed number of stripes to use for all destination files and directories. A greater number or size defined with the suffixes k, m, g, and t for KiB, MiB, GiB,

and TiB, respectively, specifies that files will be allocated one stripe per given size while directories will be striped according to the default policy. Finally, an arbitrary Perl expression (see perlsyn(1) for details) involving the constants NM, SZ, SC, and SS for source name, size, stripe count, and stripe size, respectively, may be specified to dynamically define the stripe count differently for every file and directory in the transfer. For example, the expression "NM =~ /foo/ ? 4 : (SZ < 10g ? 2g : 10g)" would set the stripe count of files whose name contains "foo" to 4, and the stripe count of files whose name does not contain "foo" to either one stripe per 2 GiB when the file size is less than 10 GiB or one stripe per 10 GiB otherwise.
Striping behavior may be further refined by specifying a stripe size expression and/or Lustre pool name expression with similar conventions. The stripe count and/or stripe size can be left empty before the colons when specifying the stripe size or pool, respectively. For example, --stripe=::4m would specify the stripe size to be 4 MiB while using the default stripe count policy and, similarly, --stripe=::::pool1 would use the pool "pool1" while using the default stripe count and stripe size. Note that if the stripe pool is a Perl expression and not a simple alphanumeric pool name, pool names must use Perl conventions for indicating strings such as quotes and/or quote-like operators, for example: "NM =~ /foo/ ? q(poolfoo) : q(poolbar)"

--threads=NUM
: Use the given number of threads in multi-threaded transports and checksum utilities (currently, mcp and msum). The default number of threads is 4. Increasing the number of threads can increase transfer/checksum performance when a host has excess resource capacity, but can reduce performance when any associated resource has reached its maximum.

--verify-fast
: By default, files are checksummed at the source and destination to verify that they have not been corrupted with the source being read once during the copy and again during the checksum. This option specifies that the source copy buffer should be reused when possible for the source checksum calculations. This potentially increases performance up to 33%, but does not allow bits corrupted during the initial read to be detected.

--window=SIZE
: Use a TCP send/receive window of the given size in TCP-based transports (currently, bbftp and fish-tcp). The suffixes k, m, g, and t may be used for KB, MB, GB, and TB, respectively. The default is the product of the specified/estimated bandwidth and the round-trip time between source and destination. Note that it is usually preferable to specify the ??bandwidth option, which allows an appropriate window size to be set automatically. Increasing the window size allows TCP to operate more efficiently over high bandwidth and/or high latency networks, but too high a value can overrun the receiver and cause packet loss.

Transfer Throttling Options

Transfers can be throttled to prevent resource exhaustion when they reach configured thresholds for CPU, disk, I/O, and/or network utilization.

--cpu=NUM
: Throttle the transfer when the local CPU usage reaches the specified percent of the total available. This option is disabled by default but may be desirable to prevent transfers from consuming too much of the local CPU. Once the given threshold is reached, a sleep period will be induced between each batch of files to achieve an average CPU utilization equal to the value specified. Note that this functionality is currently supported only on Unix-like systems.

--disk=NUM1:NUM2
: Suspend/resume the transfer when the target filesystem disk usage reaches the specified percent of the total available. This option is disabled by default but may be desirable to prevent transfers from consuming too much local or remote disk space. Once the first threshold is reached, the transfer will suspend until enough disk resources have been freed on the target to bring the disk utilization under the second threshold. Note that this functionality is currently only supported on Unix-like systems.

--io=NUM
: Throttle the transfer when the local I/O usage reaches the specified rate in MB/s. This option is disabled by default but may be desirable to prevent transfers from consuming too much of the local I/O bandwidth. Once the given threshold is reached, a sleep period will be induced between each batch of files to achieve an average I/O rate equal to the value specified.

--ior=NUM
: Throttle the transfer when the local I/O reads reach the specified rate in MB/s. This option is similar to the --io option, but only applies to reads.

--iow=NUM
: Throttle the transfer when the local I/O writes reach the specified rate in MB/s. This option is similar to the --io option, but only applies to writes.

--net=NUM
: Throttle the transfer when the local network usage reaches the specified rate in MB/s. This option is disabled by default but may be desirable to prevent transfers from consuming too much of the local network bandwidth. Once the given threshold is reached, a sleep period will be induced between each batch of files to achieve an average network rate equal to the value specified.

--netr=NUM
: Throttle the transfer when the local network reads reach the specified rate in MB/s. This option is similar to the --net option but only applies to reads.

--netw=NUM
: Throttle the transfer when the local network writes reach the specified rate in MB/s. This option is similar to the --net option but only applies to writes.

For transfers on the same host within the NAS enclave—for example, between two directories on pfe*21*—the syntax for shiftc is similar to the cp command.

Note: If source and destination paths are not specified on the command line, any number of source/destination combinations will be read from the standard input, stdin (one combination per line).

The syntax for local Shift transfers is as follows:

shiftc [*option*]... *source dest*
shiftc [*option*]... *source... directory*
shiftc [*option*]...

For information about Shift options, including those used in the examples in this article, see Shift Command Options.

Transferring Files Locally

The examples in this section show you how use the shiftc command to transfer files on the same NAS host.

Note: The first example includes output; subsequent examples show only the command line.

- Copy local *file1* into existing directory /u/*username*:

  pfe*21*% shiftc *file1* /u/*username*
  Shift id is 1
  Detaching process (use --status option to monitor progress)

- Copy *file1* from /u/*username* to /nobackup/*username*/*dir1* on pfe*21*:

  pfe*21*% shiftc /u/*username/file1* /nobackup/*username/dir1*

- Recursively copy the directory *inputs* inside another directory, /nobackup/*username/dir2*:

  % shiftc -r *inputs* /nobackup/*username/dir2*

  Complete the same operation with data verification turned off:

  % shiftc -r --no-verify *inputs* /nobackup/*username/dir2*

  TIP: Although using the --no-verify option can improve the speed of your transfer, it is not recommended because without data verification, the integrity of your data cannot be ensured.
- Copy local *file1* in the current directory to existing local directory /u/*username/dir1*:

  pfe*21*% shiftc *file1* /u/*username*/*dir1*

- Recursively copy local directory /nobackup/*username/dir1* to local directory /nobackup/*username/dir2* using 2 client hosts to perform the transfer:

  pfe*21*% shiftc -r --hosts=2 /nobackup/*username/dir1* /nobackup/*username/dir2*

- Recursively copy local directory nobackup/*username/dir1* to local directory *dir2*, but exclude files ending in .log:

  pfe21% shiftc -r --exclude='\.log$' nobackup/*username/dir1 /dir2*

Creating and Extracting Tar Files Locally

You can use Shift to transfer a directory and write it into a tar file it in one step, resulting in a portable tar file that can be read by either shiftc or tar. In the same step, you can also create a table-of-contents (.toc) file that lists the files contained in the archive along with their sizes and attributes (recommended).

Because of their sequential nature, tar files cannot be efficiently updated in place. As a workaround, incremental tar files can be used, which are separate tar files that consist of files updated after the time the original or subsequent incremental updates were created.

Creating Tar Files

The examples in this section show you how to create tar files on the same NAS host.

- Create a tar file (*dir1*.tar) of the directory /nobackup/*username/dir1* and put it in the current directory, along with a corresponding table of contents (*dir1*.tar.toc):

  pfe*21*% cd /nobackup/*username*
  pfe*21*% shiftc --create-tar --index-tar *dir1 dir1*.tar

  Note:? If the *dir1* directory is over 1 TB, it will be split into multiple tar files prefixed with *dir1*.

- The NAS nobackup filesystems are mounted on Lou. This makes it possible to create local tar files from your nobackup

filesystem directly to your Lou home directory when you are logged into Lou (lfe*5-8*).

For example, to transfer the *dir1* directory as a tar file from your nobackup filesystem to the *data_dir* directory in your Lou home directory:

```
lfe5% cd /nobackupp17/username/data_dir
lfe5% shiftc --create-tar --index-tar dir1 /u/username/data_dir/dir1.tar
```

Note: In the above example, the first step is to change into the directory /nobackupp17/*username/data_dir*, which is the parent directory of *dir1* (the directory you are transferring). This prevents the tar operation from creating extraneous prepended directories when the tar file is extracted.

Creating Incremental Tar Files

The examples in this section show you how to create incremental tar files on the same NAS host.

- Create an incremental tar file (*dir1*-2020.tar) of all files modified on or after January 1st, 2020 in the /nobackup/*username/dir1* directory, and put it in the current directory, along with a corresponding table of contents (*dir1*-2020.tar.toc):

  ```
  pfe21% cd /nobackup/username
  pfe21% shiftc --create-tar --index-tar --newer="Jan 1 2020" dir1 dir1-2020.tar
  ```

- Create an incremental tar file (*dir1*-update.tar) of all files modified in the *dir1* directory of your nobackup directory after the original tar (*dir1*.tar) was successfully completed, to the *data_dir* directory in your Lou home directory:

  ```
  lfe5% cd /nobackupp/username/data_dir
  lfe5% shiftc --create-tar --index-tar --newer=`stat -c %Y /u/username/data_dir/dir1.tar` \
  dir1 /u/username/data_dir/dir1-update.tar
  ```

  Notes:

  - The shiftc command line shown above is too long to be formatted as one line, so it is broken with a backslash (\).
  - The version of stat currently deployed on the systems does not allow file creation time to be retrieved, so the above example might miss files that were modified or created while the original tar was being written.

Extracting Tar Files

The examples in this section show you how to extract files from a tar file on the same host.

- Extract the tar file *dir1*.tar on Lou directly into your /nobackup directory:

  ```
  lfe5% cd /nobackup/username/data_dir
  lfe5% shiftc --extract-tar /u/username/data_dir/dir1.tar .
  ```

- If *dir1* was over 1 TB and therefore split into multiple tar files:

  ```
  pfe21% shiftc --extract-tar /u/username/data_dir/dir1.*tar .
  ```

- Extract the files *1g.20* through *1g.29* from *dir1*.tar to the current directory:

  ```
  pfe21% shiftc --extract-tar --include='1g\.2[0-9]' dir1.tar .
  ```

  Note: As shown in this example, shiftc uses Perl-style regular expressions for some options.

For transfers between two host systems within the NAS enclave—for example, from a PFE to an LFE—the syntax for the shiftc command is nearly the same as that of local transfers.

Note: If source and destination paths are not specified on the command line, any number of source/destination combinations will be read from stdin (one combination per line).

shiftc [OPTION]... *source dest*
shiftc [OPTION]... source... *directory*
shiftc [OPTION]...

For information about Shift options, including those used in the examples in this article, see Shift Command Options.

Transferring Files Within the Enclave

The examples in this section show you how to transfer files between two hosts within the enclave.

Note: The first example includes output; subsequent examples show only the command line.

Perform basic host-to-host transfers (for example, /u/*username*/*file1* on pfe*21* to your data/*dir2* directory on Lou):

pfe*21*% shiftc /u/*username*/*file1* lfe:data/*dir2*
Shift id is 2
Detaching process (use --status option to monitor progress)

TIP: If you have a small transfer that will complete quickly, using the --wait option will provide a helpful indication that your transfer is complete—you'll know it's done as soon as your prompt returns. Also, if you include shiftc in a script, using the --wait option will prevent the next step in the script from starting until the transfer is complete.

The next two examples show this option.

- Copy local *file1* in the current directory to your Lou home directory while waiting for completion:

  pfe*21*% shiftc --wait *file1* lfe:

- Synchronize the local directory /u/*username*/*dir1* with the Lou directory /u/*username/dir2/dir1* while waiting for completion:

  pfe*21*% shiftc -r --sync --wait /u/*username*/*dir1* lfe:/u/*username*/*dir2*

Creating and Extracting Tar Files Within the Enclave

You can use Shift to transfer a directory and write it into a tar file it in one step, resulting in a portable tar file that can be read by either shiftc or tar. In the same step, you can also create a table-of-contents (.toc) file that lists the files contained in the archive along with their sizes and attributes (recommended).

Because of their sequential nature, tar files cannot be efficiently updated in place. As a workaround, incremental tar files can be used, which are separate tar files that consist of files updated after the time the original or subsequent incremental updates were created.

Creating Tar Files

This example creates a tar file (*dir1*.tar) of the *dir1* directory in /nobackup/*user1* directly into your Lou home directory, along with a corresponding table of contents (*dir1*.tar.toc):

pfe*21*% cd /nobackup/*user1*
pfe*21*% shiftc --create-tar --index-tar *dir1* lfe:*dir1*.tar

Note: If the *dir1* directory is over 1 TB, it will be split into multiple tar files prefixed with *dir1*.

Creating Incremental Tar Files

The examples in this section show you how to create incremental tar files from one host to another within the enclave.

- Create an incremental tar file (*dir1*-2020.tar) of all files modified on or after January 1st, 2020 in the directory /nobackup/*username*/*dir1* directly to your Lou home directory, along with a corresponding table of contents (*dir1*-2020.tar.toc):

  pfe21% cd /nobackup/*username*
  pfe21% shiftc --create-tar --index-tar --newer="Jan 1 2020" *dir1* lfe:*dir1*-2020.tar

- Create an incremental tar file (*dir1*-update.tar) of all files modified in the *dir1* directory of your nobackup directory after the original tar (*dir1*.tar) was successfully completed, to the *data_dir* directory in your Lou home directory:

  pfe21% cd /nobackupp/*username*/*data_dir*
  pfe21% shiftc --create-tar --index-tar --newer=`ssh lfe stat -c %Y *dir1*.tar` *dir1* lfe:*dir1*-update.tar

Note: The version of stat currently deployed on the systems does not allow file creation time to be retrieved, so the above example might miss files that were modified or created while the original tar was being written.

Extracting Tar Files

These examples show you how to extract tar files from one host to another within the enclave.

- Extract the *dir1*.tar file from your Lou home directory to the current directory:

  pfe*21*% shiftc --extract-tar lfe:*dir1*.tar .

- If *dir1* was over 1 TB and therefore split into multiple tar files, extract them from your Lou home directory to the current directory as follows:

  pfe*21*% shiftc --extract-tar lfe:'*dir1*.*tar' .

  Note: As shown in the example, quotation marks are required when using wildcards (e.g., *) with sources on a remote host.

- Extract the files *1g.20* through *1g.29* from *dir1*.tar to the current directory:

  pfe21% shiftc --extract-tar --include='1g\.2[0-9]' lfe:*dir1*.tar .

  Note: As shown in this example, shiftc uses Perl-style regular expressions for some options.

Transfers between a remote system and a host system within the NAS enclave—for example, between your local system and a PFE—must be carried out using the Secure Unattended Proxy (SUP).

To use the SUP for Shift transfers to NAS systems, you must first download the SUP client, authorize one or more NAS hosts for SUP operations, and then authorize one or more NAS directories for writes. A brief summary of these steps is shown below. For a full overview, see Using the Secure Unattended Proxy.

Downloading SUP to Enable Remote Transfers

Complete the following steps to set up your system for remote transfers:

1. Download and install SUP client (one time):

   your_local_system% wget -O sup https://www.nas.nasa.gov/hecc/support/kb/file/9
   your_local_system% chmod 700 sup
   your_local_system% mv sup ~/bin

2. Authorize host for SUP operations (one time per host):

   your_local_system% ssh pfe*21*
   pfe*21*% touch ~/.meshrc

3. Authorize directories for writes (one or more times per host):

   your_local_system% ssh pfe*21*
   pfe*21*% echo /tmp >>~/.meshrc
   pfe*21*% echo /nobackup/$USER >>~/.meshrc
   pfe*21*% echo /u/$USER >>~/.meshrc

Performing Remote Transfers

For remote transfers, the shiftc syntax is similar to local transfers and transfers between two hosts within the enclave, except that you must prepend sup (the SUP client) to each shiftc command. Also, remote Shift transfers must always be initiated from the system that is external to the NAS enclave, but files may be transferred in either direction.

sup shiftc [OPTION]... *source dest*
sup shiftc [OPTION]... *source... directory*
sup shiftc [OPTION]...

For example, the following enclave-to-enclave transfer copies *file1* into the directory ~/data/*run2* on lfe5:

pfe21% shiftc /home/*username/file1* lfe*5*:~/data/*run2*

If the above example were changed into a remote transfer, it would become:

your_local_system% sup shiftc /home/*username/file1* lfe*5*:~/data/*run2*

In general, shiftc [*args*] becomes sup shiftc [*args*].

Note: For information about Shift options, including those used in the examples in this article, see Shift Command Options.

File Transfer Examples

The examples in this section show you how to transfer files between your local system and a host within the enclave. The first example includes output; subsequent examples show only the command line.

- Perform a remote transfer via the Secure Unattended Proxy (for example, /u/*username*/*file1* on your local system to your home directory on pfe*21*):

  your_local_system% sup shiftc /u/*username/file1* pfe*21*:
  Shift id is 3
  Detaching process (use --status option to monitor progress)

- Recursively copy local directory /u/*username/dir1* on your local system to directory /u/*username/dir2* on lfe*5*:

  your_local_system% sup shiftc -r /u/*username/dir1* lfe*5*:/u/*username/dir2*

- Recursively copy remote directory /u/*username/dir2* on lfe*5* to the current directory on your local system using an encrypted transport:

  your_local_system% sup shiftc -r --secure lfe*5*:/u/*username/dir2* .

Creating and Extracting Tar Files Remotely

You can use Shift to transfer a directory and write it into a tar file it in one step, resulting in a portable tar file that can be read by

either shiftc or tar. In the same step, we recommend that you also create a table-of-contents (.toc) file that lists the files contained in the archive along with their sizes and attributes.

Because of their sequential nature, tar files cannot be efficiently updated in place. As a workaround, incremental tar files can be used, which are separate tar files that consist of files updated after the time the original or subsequent incremental updates were created.

Creating Tar Files

This example creates a tar file (*dir1*.tar) of the directory *dir1* on your local system directly into your Lou home directory, along with a corresponding table of contents (*dir1*.tar.toc):

your_local_system% sup shiftc --create-tar --index-tar *dir1* lfe:*dir1*.tar

Note: If the *dir1* directory is over 1 TB, it will be split into multiple tar files prefixed with *dir1*.

Creating Incremental Tar Files

The example in this section shows you how to create incremental tar files from your local system to your Lou home directory.

Create an incremental tar file (*dir1*-2020.tar) of all files modified on or after January 1st, 2020 in the directory *dir1* on your local system directly to your Lou home directory, along with a corresponding table of contents (*dir1*-2020.tar.toc):

your_local_system% sup shiftc --create-tar --index-tar --newer="Jan 1 2020" *dir1* lfe:*dir1*-2020.tar

Extracting Tar Files

These examples show you how to extract tar files from your local system to your Lou home directory.

- Extract the *dir1*.tar file from your local system to your Lou home directory:

  your_local_system% sup shiftc --extract-tar lfe:*dir1*.tar .

- If *dir1* was over 1 TB and therefore split into multiple tar files:

  your_local_system% sup shiftc --extract-tar lfe:'*dir1.**tar' .

  Note: As shown in this example, quotation marks are required when using wildcards (e.g., *) with sources on a remote host.

- Extract the files *1g.20* through *1g.29* from *dir1*.tar from your local system to your Lou home directory:

  your_local_system% sup shiftc --extract-tar --include='1g\.2[0-9]' lfe:*dir1*.tar .

  Note: As shown in this example, shiftc uses Perl-style regular expressions for some options.