Use the file transfer commands scp or shiftc when the source and destination are located on different hosts—either on two different NAS high-end computing hosts, or on a NAS host and a remote host such as your local desktop system.

The basic syntax is:

copy-command [options]...source...destination

## scp Command

The Secure Copy Protocol (scp) command, based on the Secure Shell Protocol (SSH), is a means of securely transferring files between a local and a remote host. Both your authentication information (such as password or passcode) and your data are encrypted.

The most widely used scp program is from OpenSSH.

## Where is scp installed at NAS?

A copy of scp from OpenSSH without the patch is available on all front-end systems of NAS compute resources (PFEs, AFEs, CFEs, CGHFEs), Lou, and the secure front-end systems (SFEs).

## Do you need it installed on your local host?

If you have a version of SSH installed on your local host, scp is most likely already installed there.

## When to use it?

Typically, scp is used to transfer small files within NAS (<< 5 GB) or offsite (<< 1 GB) that take a reasonable amount of time to complete.

## Examples

In these examples, "outbound" means the command is initiated on a NAS compute resources front-end host or a Lou front-end host, whether the file is being pushed or pulled. "Inbound" means the command is initiated on your local host.

Note: Omit *local_username@* and *nas_username@* in the examples below if your local username and NAS username are identical. These examples assume you already know how to log into the NAS enclave.

For outbound transfer:

lou% scp *local_username@your_localhost.domain:file1 ./file2*

For inbound transfer if SSH passthrough has been set up correctly:

*your_localhost*% scp *file1 nas_username*@lou.nas.nasa.gov:*file2*

For inbound transfer if you have not set up SSH passthrough:

*your_localhost*% scp -oProxyCommand='ssh *nas_username*@sfeX.nas.nasa.gov
ssh-proxy %h' *file1 nas_username*@lou.nas.nasa.gov:*file2*

where sfeX is sfe[6-8].

Note: Due to formatting issues, the command line in the above example is shown on two lines. It should be entered as one line.

## Performance

Within the NAS secure enclave, depending on source and destination hosts and other factors, the performance range will be 40-100 MB/sec.

If your data will compress well, consider enabling compression by adding -C to your scp command line.

We recommend using OpenSSH 5.0 or a newer version.

## Shift Command (shiftc)

Shift (shiftc) is a NAS-developed tool for performing automated local and remote file transfers. Shift utilizes a variety of underlying file transports to achieve maximum performance for files of any size on any file system.

## Where is it installed at NAS?

Shift is installed on Lou and on the PFEs.

## Do you need it installed on your local host?

For transfers between your local host and NAS systems, you must install the SUP client as discussed in Shift File Transfer Overview.

## When to use it?

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).

Shift can be used as a drop-in replacement for scp between any enclave systems. For transfers between your local host and NAS systems, the transfer must be initiated from your local host with shiftc invoked via the SUP client (that is, using the command sup shiftc). If an encrypted transfer is required, use the shiftc --secure option.

## Example

pfe25% shiftc /nobackupp2/*username/filename* lou:
*your_localhost*% sup shiftc pfe:*filename* .

## Performance

Shift uses the highest performing file transport that is available on both sides of the transfer, and is optimal for the sizes of the files being transferred.

For more information, see File Transfer: Overview and Shift File Transfer Overview.

It is a good practice to confirm whether your files are complete and accurate before you transfer the files to or from NAS, and again after the transfer is complete.

The easiest way to verify the integrity of file transfers is to use the NAS-developed Shift tool for the transfer, with the --verify option enabled. As part of the transfer, Shift will automatically checksum the data at both the source and destination to detect corruption. If corruption is detected, partial file transfers/checksums will be performed until the corruption is rectified.

For example:

pfe*21*% shiftc --verify $HOME/*filename* /nobackupp*X*/*username*
lou% shiftc --verify /nobackupp*X*/*username/filename* $HOME
*your_localhost*% sup shiftc --verify *filename* pfe:

In addition to Shift, there are several algorithms and programs you can use to compute a checksum. If the results of the pre-transfer checksum match the results obtained after the transfer, you can be reasonably certain that the data in the transferred files is not corrupted. If data *is* corrupted during a transfer, a good checksum algorithm will yield different results before and after the transfer, with high probability.

The following checksum programs are available on HECC systems:

sum
: Computes a checksum using the BSD sum or System V sum algorithm; also counts the number of blocks (1 KB-block or 512 B-block) in a file

cksum
: Computes a cyclic redundancy check (CRC) checksum; also counts the number of bytes in a file

md5sum
: Computes a 128-bit MD5 checksum, which is represented by a 32-character hexadecimal number

msum
: High performance multi-threaded checksum utility, developed at NAS. By default, computes 128-bit MD5 checksums, but you can compute other types using --hash-type. Note that for full compatibility with md5sum output, you must use the --split-size=0 option, which will also decrease performance on large files. For more information, see **man msum** on any HECC front-end system.

For example:

% ls -l *filename*
-rw------- 1 *username group_id* 67358 Nov 15 11:49 *filename*

% sum *filename*
50063 66

% cksum *filename*
269056887 67358 *filename*

% md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

% msum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

To check the integrity of an existing tar file against a directory:

% tar -tf *dir_name*.tar | sort > sums.*dir_name*.tar
% find *dir_name* | xargs md5sum | sort>sums.*dir_name*.dir
% diff sums.*dir_name*.tar sums.*dir_name*.dir

The md5sum utility is more reliable than the sum or cksum commands for detecting accidental file corruption, as the chances of accidentally having two files with identical MD5 checksums are extremely small. The utility is installed by default in most Unix, Linux, and Unix-like operating systems. We recommend that you compute the md5sum of a file before and after the transfer.

The following example shows that the file *filename* is complete and accurate after the transfer, based on its md5sum.

pfe*21*% md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

pfe*21*% scp *filename local_username@your_localhost*:

*your_localhost*%md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

See the **sum, cksum, md5sum, and msum man pages** for more information on these commands.

Encryption helps protect your files during inter-host file transfers that use protocols that are not already encrypted—for example, when using ftp or when using shiftc without the --secure option. We recommend using the GNU Privacy Guard (GPG), an Open Source OpenPGP-compatible encryption system.

GPG has been installed on all NAS compute systems and Lou in the /usr/bin/gpg directory. If you do not have GPG installed on the system(s) that you would like to use for transferring files, please see the GPG website.

## Choosing What Cipher to Use

We recommend using the cipher AES256, which uses a 256-bit Advanced Encryption Standard (AES) key to encrypt the data. Information on AES can be found at the National Institute of Standards and Technology's Computer Security Resource Center.

You can set your cipher in one of the following ways:

- Add --cipher-algo AES256 to your ~/.gnupg/gpg.conf file.
- Add --cipher-algo AES256 in the command line to override the default cipher, CAST5.

## Examples

If you choose not to add the cipher-algo AES256 to your gpg.conf file, you can add --cipher-algo AES256 on any of these simple example command lines to override the default cipher, CAST5.

## Creating an Encrypted File

Both commands below are identical. They encrypt the test.out file and produce the encrypted version in the test.gpg file:

```
% gpg --output test.gpg --symmetric test.out
```

```
% gpg -o test.gpg -c test.out
```

You will be prompted for a passphrase, which will be used later to decrypt the file.

## Decrypting a File

The following command decrypts the test.gpg file and produces the test.out file:

```
% gpg --output test.out -d test.gpg
```

You will be prompted for the passphrase that you used to encrypt the file. If you don't use the --output option, the command output goes to STDOUT. If you don't use any flags, it will decrypt to a file without the .gpg suffix. For example, using the following command line would result in the decrypted data in a file named "test":

```
% gpg test.gpg
```

## Selecting a Passphrase

Your passphrase should have sufficient information entropy. We suggest that you include five words of 5-10 letters in size, chosen at random, with spaces, special characters, and/or numbers embedded into the words.

You need to be able to recall the passphrase that was used to encrypt the file.

## Factors that Affect Encrypt/Decrypt Speed on NAS Filesystems

We do not recommend using the --armour option for encrypting files that will be transferred to/from NAS systems. This option is mainly intended for sending binary data through email, not via transfer commands such as ftp or shiftc with the -secure option. The file size tends to be about 33% bigger than without this option, and encrypting the data takes about 10-15% longer.

The level of compression used when encrypting/decrypting affects the time required to complete the operation. There are three options for the compression algorithm: none, zip, and zlib.

- --compress-algo none or --compress-algo 0
- --compress-algo zip or --compress-algo 1
- --compress-algo zlib or --compress-algo 2

For example:

```
% gpg --output test.gpg --compress-algo zlib --symmetric test.out
```

If your data is not compressible, --compress-algo 0 (none) gives you a performance increase of about 50% compared to --compress-algo 1 or --compress-algo 2.

If your data is highly compressible, choosing the zlib or zip option will not only increase the speed by 20-50%, it will also reduce the file size by up to 20x. For example, in one test on a NAS system, a 517 megabyte (MB) highly compressible file was compressed to 30 MB.

The zlib option is not compatible with PGP 6.*x*, but neither is the cipher algorithm AES256. Using the zlib option is about 10% faster than using the zip option on a NAS system, and zlib compresses about 10% better than zip.

## Random Benchmark Data

We tested the encryption/decryption speed of three different files (1 MB, 150 MB, and 517 MB) on NAS systems. The file used for the 1 MB test was an RPM file, presumably already compressed, since the resulting file sizes for the none/zip/zlib options were within 1% of each other. The 150 MB file was an ISO file, also assumed to be a compressed binary file for the same reasons. The 517 MB file was a text file. These runs were performed on a CXFS filesystem when many other users' jobs were running. The performance reported here is for reference only, and not the best or worst performance you can expect.

| **Using AES256 as the Cipher Algorithm** | | | |
|---|---|---|---|
| | **1 MB File** | **150 MB File** | **517 MB File** |
| **with --armour** | ~5.5 secs to encrypt | ~40 secs to encrypt | |
| **without --armour** | ~4 secs to encrypt | ~35 secs to encrypt | |
| **without --armour, zlib compression** | | ~33 secs to encrypt; ~28 secs to decrypt to file | ~33 secs, resultant file size ~30 MB; ~34 secs to decrypt to file |
| **without --armour, zip compression** | | ~36 secs to encrypt; ~31 secs to decrypt to file | ~38 secs, resultant file size ~33 MB; ~34 secs to decrypt to file |
| **without --armour, no compression** | | ~19 secs to encrypt; ~25 secs to decrypt to file | ~49 secs, resultant file size ~517 MB; ~75 secs to decrypt to file |

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.

## Using SUP for Remote Transfers

## Using the Secure Unattended Proxy (SUP)

UPDATE: As of June 2, 2023, you must ensure that your SUP client has been updated with the latest version 8.13 (available below). You must do one of the following to replace their SUP client manually:

1. Follow the installation instructions for the SUP client.
   or
2. Run the following command directly on top of the old client, or to a new file:
   ssh sup.nas.nasa.gov mesh-update --file=mc>sup
   If downloading the client to a new location, use chmod 700 sup to make the client executable, then move it to a location in your $PATH.

The Secure Unattended Proxy (SUP) allows you to perform remote operations on specific hosts within the NAS enclave front-ends (currently, PFEs and LFEs) *without* needing to use your RSA SecurID token at the time of the operation.

To accomplish this, you must first obtain special "SUP keys" using RSA SecurID authentication, which you can then use to perform operations from unattended jobs and/or scripts. Each SUP key is valid for a period of *one week* from the time it is generated. You may have multiple SUP keys at the same time, which will expire asynchronously.

SUP keys are currently allowed to call scp, sftp, qstat, rsync, shiftc, ssh-balance, and test.

Note: The shiftc command is built into the sup command; therefore, when you download the SUP client, you will also get shiftc.

## SUP Usage Summary

The steps in this section demonstrate how to quickly get up and running with the SUP. Each step is explained in more detail in subsequent sections.

In these steps, *host* refers to a PFE or LFE; the command lines shown are examples.

1. Download and install the SUP client in your personal bin directory (one time).

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

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

   *pfe*% touch ~/.meshrc

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

   *pfe*% echo /nobackupp2 >> ~/.meshrc

4. Execute a command (each time)

   *your_localhost*% sup scp *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

5. Examine expected output (as needed)
6. Troubleshoot problems (as needed)

## SUP Client

The SUP client performs all the steps needed to execute commands through the SUP as if the SUP itself did not exist. Commands that are allowed to pass through the SUP can be executed as if the remote host were directly connected by simply prepending the client command sup, as shown in example 4, above. Besides executing remote commands, the client also includes an operating-system-independent virtual filesystem that allows files across all SUP-connected resources to be accessed using standard filesystem commands.

## Requirements

The SUP requires OpenSSH version 5.4 (or later) and Perl version 5.8.5 (or later) and has been tested successfully on Linux, OS X, and Windows under Cygwin. These requirements are typically installed by default on Linux and OS X, but may need to be manually installed on Cygwin through the package manager as the openssh and perl packages. For general Cygwin installation instructions, see Installing Cygwin (PDF).

## Installation

Complete the following steps to install the client:

1. Download the client and save it to a file called *sup*
2. Make the client executable by using chmod 700 sup
3. Move the client to a location in your $PATH

## SSH Configuration

If your local username differs from your NAS username, it is recommended that you add the following to your~/.ssh/config file, where *nas_username* is replaced with your NAS username:

```
Host sup.nas.nasa.gov sup-key.nas.nasa.gov
    User nas_username
```

Note: If you are using a config file based on the NAS config template, you do not have to do this step.

Alternatively, the client's -u option can be used as described in the next section. If your local username is the same as your NAS username, no additional configuration or command-line options are required.

## SUP Command-Line Options

-b
: Disable user interaction; for use within scripts. Note that the client will fail if any interaction is required--normally only needed when your SUP key has expired or is otherwise unavailable.

-u *username*
: Specify NAS username. Note that this option is required if your local username differs from your NAS username and you have not modified your SSH configuration appropriately.

-v
: Enable verbose output for debugging purposes.

## SUP Authorizations

The basic set of operations that may be performed using the SUP is specified by the administrator. To protect accounts from malicious use of SUP keys, users must grant execute and write permissions to SUP operations on each target system.

## Authorizing Execute Operations

By default, even SUP operations permitted by site policy are not allowed to execute on a given host. To enable SUP operations to a given host (PFE or LFE), the file ~/.meshrc must exist on that host, which can be created by invoking the following:

```
% touch ~/.meshrc
```

Note that the PFEs share their home filesystems, so this must only be done on one of these nodes. Other systems must be authorized separately. Once this file exists on a host, all operations permitted by site policy are allowed to execute on that host.

## Authorizing Write Operations

By default, SUP operations are not allowed to write to the filesystem on a given host. To enable writes to a given directory on a given host, that directory must be added (on a separate line) to the ~/.meshrc file on that host. For example, the following lines in ~/.meshrc indicate that writes should be permitted to /nobackupp2 and your home directory.

```
/nobackupp2
/u/username
```

Each directory is the root of allowed writes, so this configuration would allow writes to all files and directories rooted at /nobackupp2 and your home directory (for example, /nobackupp2/*some/dir*).

Note that the root directory cannot be authorized. Also note that dot files (that is,~/.*) in your home directory are never writable regardless of the contents of ~/.meshrc.

## Executing Commands Through SUP

Usage examples of each command that may be executed through the SUP are given below.

Note: SUP commands must be authorized for execution on each target host, and that transfers to a given host must beauthorized for writes. Before a given operation is performed, the client may ask for certain information, including the existing or new passphrase for ~/.ssh/id_rsa, the password + passcode for sup.nas.nasa.gov, and/or the password + passcode for sup-key.nas.nasa.gov.

For more detailed information on each command, see their corresponding **man pages**.

## File Transfer Commands

**rsync**

*your_localhost*% sup rsync *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

If you intend to transfer files to your home directory, note that even if your home directory has beenauthorized for writes, rsync transfers to your home directory will fail unless the --inplace option is specified. This is because rsync uses temporary files starting with "." during transfers, which cannot be written in your home directory. You can avoid this problem by specifying --inplace as shown in the following example:

*your_localhost*% sup rsync --inplace *foobar* pfe*21*:

**scp**

*your_localhost*% sup scp *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

**sftp**

*your_localhost*% sup sftp pfe*21*

**shiftc**

*your_localhost*% sup shiftc *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

For more information, see Using Shift for Remote Transfers and Tar Operations.

## File Monitoring Command

**test**

*your_localhost*% sup ssh pfe*21* test -f /u/*username*/c_*foobar*

## Job Monitoring Command

**qstat**

*your_localhost*% sup ssh pfe*21* qstat @pbspl1

## ssh-balance and bbFTP

**ssh-balance**

*your_localhost*% sup ssh-balance -l pfe

Note that this command allows the load balancer among the PFEs to be used with bbftp:

*your_localhost*% sup bbftp -e "put *foobar* /u/*username*/*c_foobar*" `sup ssh-balance -l pfe`

## SUP Expected Output

The following sequence shows the expected output for the command:

*your_localhost*% sup scp *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

for a user who has never used the SUP before.

The conditions under which each sub-sequence will be seen are indicated next to each header. Most of the items will only be seen once or during key generation. A second invocation will only show the command output portion.

1. **Identity initialization** (seen once per identity)

   Initializing identity on sup-key.nas.nasa.gov (provide login information)
   (username@sup-key.nas.nasa.gov) Password:
   (username@sup-key.nas.nasa.gov) Enter PASSCODE:
   key SHA256:p3VqOcfV46fJwhOpYs/2h2sztRkNZ3nFu6VQbl2pgfI uploaded successfully

2. **SUP key generation** (seen when no valid SUP keys available)

   Generating key on sup.nas.nasa.gov (provide login information)
   (username@sup.nas.nasa.gov) Password:
   (username@sup.nas.nasa.gov) Enter PASSCODE:

3. **Client upgrade** (seen during key generation when new client available)

   A newer version of the client is available (8.04 vs. 8.1)
   ...do you wish to replace the current version? (y/n) **y**

4. **Command output** (always seen)

   *foobar* 100% 5 0.0KB/s 00:00

## SUP Troubleshooting

The following error messages may be encountered during your SUP client usage. Note that the-v option can be given to the SUP client to output additional debugging information.

- "WARNING: Your password has expired"

  This message indicates that your current password has expired and must be changed. To change your password, you must log

in to an LDAP host (for example, Lou) through an SFE and change your LDAP password. This change will be automatically propagated to the SUP within a few minutes.

- "Permission denied (~/.meshrc not found)"

  This message indicates that you have not created a .meshrc file in your home directory on the target host. SUP commands must be authorized for execution on each target host.

- "Permission denied (unauthorized command)"

  This message indicates that you have attempted an operation that is not currently authorized by the SUP. Check that the command line is valid and that the attempted command is one of the authorized commands. Certain options to authorized commands may also be disallowed, but these should never be needed in standard usage scenarios.

- "Permission denied during file access" (various forms)

  These messages indicates that you attempted to read or write a file for which such access is not allowed. The most common cause is forgetting to authorize directories for writes. Reads and writes of ~/.* are never permitted.

- "Permission denied (publickey)"

  This message indicates that you may not have proper permissions on your ~/.ssh and/or home directory on the target host. Check to make sure that ~/.ssh is not readable/writable by other users/groups and that your home directory is not writable by other users/groups.

The SUP virtual file system (VFS) capability can be used to issue file-related commands to all SUP-connected hosts from the command line of your local host.

## Accessing Files Across Multiple Hosts

The SUP client includes a virtual file system capability that allows files across all SUP-connected resources to be accessed using standard file system commands. For example, once you have activated VFS as described below, the command:

% ls pfe*21*:/tmp

from your local host would list the files in /tmp on pfe*21*. The command:

% cp *filename* pfe*21*:/tmp

from your local host would copy the file *filename* from your current directory on your local host to /tmp on pfe*21*.

The set of supported commands includes cat, cd, chgrp, chmod, chown, cmp, cp, df, diff, du, file, grep, head, less, ln, ls, mkdir, more, mv, pwd, rm, rmdir, tail, tee, test, touch, and wc.

Note that this functionality is not a true file system since only these commands are supported and only when used from within a shell. Unlike more general approaches such as FUSE, however, the SUP capability is completely portable and can be enabled with no additional privileges or software.

Commands through the VFS functionality can act on any combination of local and remote files, where remote files are prefixed with *hostname*:. For example, the command:

% cat pfe*21*:/tmp/*rfile* ~/*lfile*

would print the file *rfile* in /tmp on pfe*21* as well as the file *lfile* in the user's home directory on the local host to the terminal. Any number of hosts can be included in any command. For example, the command:

% diff lfe2:/tmp/*lfe_file* pfe*21*:/tmp/*pfe_file*

would show the differences between the file *lfe_file* in /tmp on lfe2 and the file *pfe_file* in /tmp on pfe*21*. The client determines if any remote access is needed based on the path(s) given. If not, it will execute the command locally as given as rapidly as possible. Fully local commands also support all options with the exception of options of the form -f *value* (that is, single-dash options that take values).

## VFS Activation

## Requirements

Currently, SUP VFS functionality is only supported for bash, but csh support is planned for the future. This functionality requires Perl version 5.8.5 (note that this is more recent than version 5.6.1 required by the basic client functionality). It also requires the standard Unix utilities cat, column, false, sort, and true; and has been tested successfully on Linux, OS X, and Windows underCygwin. Note that users of Windows under Cygwin may need to install the coreutils and util-linux packages to obtain these utilities.

## Activation and Deactivation on Your Local Host

Complete these steps.

1. Install the SUP client if you have not already done so.
2. Activate VFS functionality in a bash shell.
    - For an interactive bash shell, run:

      ```
      eval `sup -s bash`
      ```

    - For a non-interactive bash shell, begin the script with the following:

      ```
      #!/bin/bash
      shopt -s expand_aliases
      eval `sup -s bash`
      ```

    The instructions will load aliases and functions that will intercept specific commands and replace them with commands that will perform the requested actions through the SUP client.

To deactivate VFS functionality in a bash shell or bash script at any time, run:

% eval `sup -r bash`

## Command-Line Options

The behavior of the virtual file system can be modified using various options at the time it is activated.

*-ocmd=opts*
: Specify default options for a given command since the VFS functionality overrides any existing aliases for its supported set of commands.

-t *transport*
: Change the file transport from its sftp default to transport. Currently, the only additional transport available is bbftp. Note that using bbftp as the transport may slow down certain operations on small files as bbftp has higher startup overhead.

-u *user*
: Specify NAS user name. Note that this option is *required* if your local user name differs from your NAS user name.

For example, the following invocation activates the client virtual file system using bbftp as the transport mechanism, *nasuser* as the user and adds colorization of local file listings using the Linux ls --color=always option.

```
% eval `sup -s bash -t bbftp -u nasuser -ols=--color=always`
```

## VFS Caveats

The VFS functionality is still somewhat experimental. In general, it works for the most common usage scenarios with some caveats. In particular:

- "Whole file" commands (that is, commands that must process the entire file), including cat, cmp, diff, grep, wc (and currently more/less due to implementation) retrieve files first before processing for efficiency. Thus, these commands should not be executed on very large files.
- There is a conflict between commands that take piped input and the custom globbing of the client, thus these commands have portions of globbing support disabled. These commands are grep, head, less, more, tail, tee, and wc. In these cases, globbing will work for absolute prefixes, but not relative. For example, grep *filename* pfe*21:*/tmp/* will work, but cd pfe*21:*/tmp; grep *filename* * will not.
- Redirection to/from remote files doesn't work. The same effect can be achieved using cat and tee (for example, grep localhost < pfe*21:*/etc/hosts > pfe*21:*/tmp/a could be done with cat pfe*21:*/etc/hosts | grep localhost | tee pfe*21:*/tmp/a > /dev/null). Redirection still works normally for local files.
- The first time a command is run involving a particular host, a SFTP connection is created to that host. When runningps, it may appear as if a zombie client process is running.

## VFS Commands

Currently supported commands and their currently supported options are below. Unsupported options will simply be ignored except where noted. All commands are still subject to SUP authorizations, thus something that cannot be executed or written normally through the SUP cannot be executed or written through this functionality either.

- cat (no options)
- cd (no options)
  Note that when changing to remote directories, cd only changes $PWD, so to make changes visible the working directory (that is, \w in bash) must be in your prompt. For example, the following prompt:

  ```
  export PS1="\h[\w]> "
  ```

  would display the current host name followed by the current working directory.
- chgrp (no options)
  Groups may be specified either by number or by name. Names will be resolved on the remote host.
- chmod (no options)
  Modes must be specified numerically (for example, 0700). Symbolic modes, such as a+rX, are not currently supported.
- chown (no options)
  Users and groups may be specified either by number or by name. Names will be resolved on the remote host.
- cmp (all options)
- cp [-r]
  Note that copies between two remote hosts transfer files to the local host first since the SUP does not allow third party transfers. Thus, very large file transfers between remote systems should be achieved using an alternate approach.
- df [-i]
  Note that 1024-byte blocks are used.
- diff (all options)
- du [-a] [-b] [-s]
  Note that 1024-byte blocks are used.
- file (all options)
- grep (all options)
- head [-*number*]
  Note that head does not support the form -n *number*, so, for example, to display the first 5 lines of a file, use -5 and not -n 5.
- less (all options)
- ln [-s]
  Note that hard links are not supported. Links from remote files to local files (for example, ln -s pfe*21:*/*filename* /*filename*) will be dereferenced during certain operations (for example, cat /*filename* will cat pfe*21:*/*filename*).
- ls [-1] [-d] [-l]

For efficiency purposes, ls behaves slightly differently for remote commands than for local commands. In particular ls -l will not show links by default and will show what is actually linked instead of the link itself. Link details can be obtained using the -d option (for example, ls -ld *).

Also for efficiency, ls processes remote files before local files, so output ordering may be changed when remote and local files are interleaved on the ls command line. For example, ls /*file1* pfe*21*: /*file2* would show pfe*21*: first, then /*file1*, then /*file2*.

- mkdir (no options)
- more (all options)
- mv (no options)
- pwd (no options)
- rm [-r]
- rmdir (no options)
- tail [-*number*]
  Note that tail does not support the form -n *number*, so, for example, to display the last 5 lines of a file, use -5 and not -n 5.
- tee [-a]
- test [-b] [-c] [-d] [-e] [-f] [-g] [-h] [-k] [-L] [-p] [-r] [-s] [-S] [-u] [-w]
  Note that compound and string tests are not supported. Compound and string tests can be achieved using multiple test commands separated by shell compound operators. For example, instead of

  % test -f pfe*21*:/*filename* -a "*abc*" != "*123*"

  do

  % test -f pfe*21*:/*filename* && test "*abc*" != "*123*"

  Alternatively, the actual test command can be executed through the SUP:

  % sup ssh pfe*21* test -f /*filename* -a "*abc*" != "*123*"

- touch (no options)
- wc (all options)

## Examples

## Inbound File Transfer Through SFEs: Examples

Inbound file transfers through the secure front ends (SFEs) require RSA SecurID token authentication. Files cannot be transferred directly to the SFEs, so transfers must be done using either SSH passthrough or using scp with the -oProxyCommand option.

Note: While this article covers file transfers through the SFEs only, you can also transfer files through the Secure Unattended Proxy.

To simplify the instructions, the approaches are described in terms of transfers to or from one of the NAS compute resources front ends, such as pfe21, but they also apply to any of the other systems that are in the secure enclave—such as other PFEs, or the Lou front ends (LFEs).

For some of the methods described, two commands are provided. The first command (a) is used if you have identical usernames on your local system and on the NAS systems, or if the usernames are different but you have set up your local ~/.ssh/config file to include the NAS username. To learn how to set this up, download the NAS **ssh_config.txt** template. The second command (b) is used if your usernames are different and you do not include the NAS username in your local ~/.ssh/config file.

### File Transfers Using scp with -oProxyCommand

The scp command is not recommended for files over 1 GB; see Remote File Transfer Commands for a comparison of commands. If you have not set up SSH passthrough, you must use scp with the -oProxyCommand option for inbound file transfers through the SFEs.

Note: Because of a formatting issue, the commands in this section are broken into two lines. Each should be on only one line.

Using scp with the -oProxyCommand option to *push* files *out* of your local system:

(a) *your_local_system*% scp -oProxyCommand='ssh sfe6.nas.nasa.gov
ssh-proxy %h' *filename* pfe21.nas.nasa.gov:

or

(b) *your_local_system*% scp -oProxyCommand='ssh *nas_username*@sfe6.nas.nasa.gov
ssh-proxy %h' *filename nas_username*@pfe21.nas.nasa.gov:

Using scp with the -oProxyCommand option to *pull* files *into* your local system:

(a) *your_local_system*% scp -oProxyCommand='ssh sfe6.nas.nasa.gov
ssh-proxy %h' pfe21.nas.nasa.gov:*filename* .

or

(b) *your_local_system*% scp -oProxyCommand='ssh *nas_username*@sfe6.nas.nasa.gov
ssh-proxy %h' *nas_username*@pfe21.nas.nasa.gov:*filename* .

### File Transfers Using SSH Passthrough

If you have set up SSH passthrough correctly, you can use scp or shiftc to transfer files between your local system and a NAS host. You can transfer files directly into the NAS host and avoid the need to double-authenticate. The passage through the SFEs is transparent.

### Using scp

Using scp to *push* files *out* of your local system:

(a) *your_local_system*% scp *filename* pfe*21*.nas.nasa.gov:

or

(b) *your_local_system*% scp *filename nas_username*@pfe*21*.nas.nasa.gov:

Using scp to *pull* files *into* your local system:

(a) *your_local_system*% scp pfe*21*.nas.nasa.gov:*filename* .

or

(b) *your_local_system*% scp *nas_username*@pfe*21*.nas.nasa.gov:*filename* .

### Using shiftc

See Shift File Transfer Overview for shiftc examples.

## More About File Transfers

See the list of links in the File Transfer Overview.

## Outbound File Transfer Examples

You can transfer files between NAS and your local site either by running the transfer command on a PFE or LFE, or by running the command on your local system. Running the command on your local system requires you to go through a secure front end (SFE) or to set up the Secure Unattended Proxy (SUP) or SSH passthrough. Therefore, if your local system is set up to allow direct inbound connections, then starting the transfer from a PFE or LFE will be much simpler than starting the transfer from your local system.

The sample command lines shown below demonstrate how to run the scp or shiftc commands on a PFE to transfer files to or from you local system. You can also apply the commands to other systems in the enclave, such as the LFEs.

Notes:

- The sample command lines use pfe21 as an example.
- Two command lines are provided for each transfer method. Use the first one if your username for your local system is the same as your username for the NAS systems. If the usernames are different, use the second.

### Using scp

To push files *out* of a PFE to your local system:

pfe*21*% scp *filename your_local_system*:
pfe*21*% scp *filename local_username@your_local_system*:

To pull files *into* a PFE from your local system:

pfe*21*% scp *your_local_system:filename* .
pfe*21*% scp *local_username@your_local_system:filename* .

### Using shiftc

In most cases, it is recommended to initiate remote transfers on the non-NAS side using SUP/Shift since all authentication, firewalls, and shift options have been preconfigured to work correctly. Remote transfers initiated from within NAS, however, are possible if the ssh port of the remote system is accessible and supports the publickey authentication method. The private key must either be loaded in your ssh-agent process or you must use a passphrase-less private key file with --identity set to its full path name. In addition, the file "shift-aux" (obtainable from /usr/local/bin on any pfe or lfe) must be in the default $PATH on the remote system. It is recommended to restrict the list of remote transports to rsync, fish, and shift unless you know which ports are accessible on the remote host and can specify --ports (for either fish-tcp or bbftp) and/or bbftpd is installed on the remote host (for bbftp).

To push files *out* of a PFE to your local system:

pfe*21*% shiftc --remote=rsync,fish,shift *filename your_local_system*:
pfe*21*% shiftc --user=*local_username* --identity=~/.ssh/*your_private_key filename your_local_system*:

To pull files *into* a PFE from your local system:

pfe*21*% shiftc --remote=rsync,fish,shift *your_local_system:filename* .
pfe*21*% shiftc --user=*local_username your_local_system:filename* .

See Shift Command Options for all available options. Transfers may be monitored and stopped/restarted using normal shift methods. See Checking Shift Transfer Status and Restarting Transfers for more information.