IBM Aspera Desktop Client Admin Guide 3.9.1

PowerLinux

Revision:1978 Generated:04/05/2019 10:09

Contents

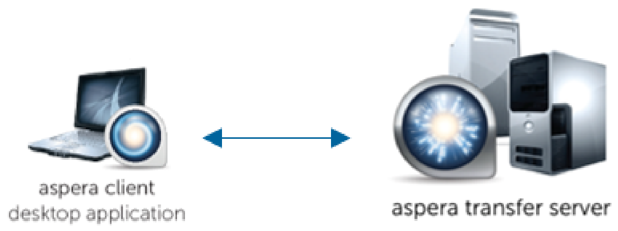
Introduction

Thanks for choosing Aspera and welcome to the world of unbelievably fast and secure data transfer.

The Basics

Aspera high-speed transfers begin when an Aspera client authenticates to an Aspera server and requests a transfer. If the client user has authorization, then transfer tools are launched on the client and server and the transfer proceeds.

For example, an IBM Aspera Desktop Client user connects to an IBM Aspera High-Speed Transfer Server and initiates a transfer:



Depending on the user's transfer request, files and folders can be transferred to the server from the client (uploaded) or transferred to the client from the server (downloaded). The source and destination can be cloud storage, an NFS or CIFS mount, and IBM Spectrum Scale storage, to name a few.

What is the Server?

The Aspera server receives transfer requests from Aspera clients, determines if the user has permission to access the server and authorization to the target area of the file system (source or destination with read or write access), and participates in transfers. The server can be:

- an on-premises installation of HST Server, IBM Aspera High-Speed Transfer Endpoint (which permits one client connection),
- a HST Server installed as part of IBM Aspera Faspex, or
- an HST Server deployed in object storage as an IBM Aspera On Demand instance, an IBM Aspera on Cloud transfer service node, or an IBM Aspera Transfer Cluster Manager node.

What is the Client?

The Aspera client is the program that requests a transfer with the Aspera server. Aspera applications that can act as clients include:

- Desktop Client,
- IBM Aspera Drive,
- IBM Aspera Connect,
- IBM Aspera Command-Line Interface,
- HST Server and HST Endpoint

What is FASP?

At the heart of your Aspera ecosystem are the FASP transfer engines Ascp and Ascp 4. Ascp maximizes data transport over any network and is particularly suited to large files. It is a powerful command-line tool and also drives transfers started in the GUI.

Ascp 4 is another command-line transfer tool that is optimized for both large files and transfers of thousands to millions of small files, handling large amounts of file metadata as part of the high-speed transfer.

Both Ascp and Ascp 4 are installed and enabled with your installation of HST Server, HST Endpoint, and Desktop Client.

The Aspera Transfer Server

Your Aspera transfer server is a powerful, customizable hub for your high speed transfer activity. Configuration settings allow you to control which clients have access for uploading or downloading data, how much bandwidth their transfers can use, the priority of those transfers, and how data is secured during and after transfer. The transfer queue can be managed on the fly, enabling you to adjust as priorities change. You can also monitor transfers and receive email notifications when transfer sessions or individual file transfers start and stop.

The Aspera Server GUI

The Aspera desktop GUI is primarily a client transfer tool, but it also offers a user-friendly interface for managing users and configuring your server on supported platforms (Windows, Linux, macOS). Security settings, bandwidth use policies, and file handling rules can all be set in the GUI. Configurations can be applied to all users (globally), to groups, or to individual users.

HST Server Web Portal

Your HST Server can be made even more accessible to clients by hosting a web-based storage directory. Authorized clients can browse files by using any modern web browser, and transfer using the free, automatically-installed Connect.

Asconfigurator: The Aspera Configuration Tool

If you are unfamiliar with the XML formatting required for your Aspera server's configuration file, you can edit your configuration with confidence by using `asconfigurator`. These commands ensure that the XML structure is correctly maintained when you add or change settings.

Tap into the Aspera Ecosystem

If you have a variety of data storage systems and internal and external customers who need access to the content in that storage, HST Server can be incorporated into a scalable Aspera data transfer ecosystem that meets your needs. Your Aspera server can be monitored and managed by IBM Aspera Console, and added as a node to IBM Aspera Faspex, IBM Aspera Shares, IBM Aspera on Cloud, and IBM Aspera Application for Microsoft SharePoint.

The Aspera Client Transfer Tools

Your installation includes the following transfer tools, some of which require an additional license for activation.

The FASP Transfer Engines: `ascp` and `ascp4`

These command line tools enable you to run transfers to any server to which you have access, and to customize the transfers (within the parameters set by the server). They are scriptable, supporting unattended data transfer and custom pre- and post-transfer file processing.

Hot Folders: Automatic Data Transfer in the GUI

Sending or receiving files can be even easier and faster by using Hot Folders. Available only on Windows, you can set up a Hot Folder to watch for and automatically transfer any new files that are added to that folder. Automatically send files to a server as they are added to a folder on your own desktop, or receive files as they are added to a folder on the server. Transfers use Ascp and are easily managed from the GUI.

Watch Folders: Automatic Content Delivery at Any Scale

Using asperawatchd and Watch Folders creates a powerful, efficient file system monitoring and automatic transfer tool that can comfortably handle millions of files and "growing" sources. Automatically transfer files as they are added to a source folder. With a REST API interface, you have full programmatic control for custom, automatic transfer processing.

Watch Folders offer the same transfer and bandwidth management options as `ascp`, and can be monitored and managed through Console. Watch Folders are enabled in your HST Server or HST Endpoint.

IBM Aspera Sync: Directory Synchronization at the Speed of FASP

When everyone needs to see the same files or you need to be sure that every file is replicated, Aspera Sync provides a high-speed tool to do it. Unique among Aspera's transfer tools, Aspera Sync supports bidirectional synchronization for optimum collaboration and consistency between computers.

Aspera Sync uses efficient file system monitoring and change detection to minimize redundant data transfer and to reduce database storage requirements. Aspera Sync offers the same transfer and bandwidth management options as `ascp`, and can be monitored and managed through Console.

Aspera Sync is installed with your HST Server and HST Endpoint, but both the client and server require a Aspera Sync-enabled license.

What's New?

DEPRECATION NOTICES

The "File Pre- and Post-Processing (Prepost)" feature will be deprecated from HST Server and HST Endpoint in versions 4.0 and onward. At that time, customers should use Inline File validation with Lua () or an External URL validator () for pre-post processing features.

DOCUMENTATION UPDATES

- Corrected Async `--transfer-threads` option example to use only numeric values in bytes.

Get Started as a Transfer Client

Aspera transfer clients connect to a remote Aspera transfer server and request a transfer with that server. Your Aspera application can be used as a client to initiate transfers with Aspera servers, as described in the following steps.

1. Review the system requirements and install Desktop Client.

   See Requirements on page 7 and Installing Desktop Client on page 7.
2. Configure the firewall.

   See Configuring the Firewall on page 8.
3. Test a locally-initiated transfer to the Aspera demonstration server to confirm your installation and firewall configuration are operational.

   For instructions, see Testing a Transfer on page 8. This provides a simple walk through of how to set up a connection with a server and transfer.
4. If you need to authenticate to the remote server with an SSH key, create an SSH key and send the public key to the server admin.

   For instructions on creating an SSH key, see Creating SSH Keys on page 42.
5. To run transfers from the command line, review the instructions for the Aspera command line client.

   Your Aspera product comes with two command line clients: `ascp` and A4. They are similar but have different capabilities. For a comparison, see Comparison of Ascp and Ascp 4 Options on page 47.

   - For more information about `ascp`, see Ascp Command Reference on page 9 and Ascp General Examples on page 24.
   - For more information about A4, see Ascp 4 Command Reference on page 53 and Ascp 4 Examples on page 61.

Once you confirm that you can transfer with your server, your basic set up is complete.

Installation and Upgrades

Before you install the current release, review the following information about hardware and software requirements, system preparation for upgrades or downgrades, installation instructions, and product security configuration.

Requirements

System requirements for Desktop Client.

- Product-specific Aspera license file.
- Ubuntu 16.04.2 LTS; Kernel: Linux 4.4.0-116-generic; architecture: ppc64-le

  Note: Your OS version must support Little Endian (LE) and it must run on IBM Power hardware that supports LE.
- SSH Server. Version 7.0 or higher is recommended.

Installing Desktop Client

To install Desktop Client, log into your computer with root permissions.

1. Download the HST Server installer.

   Use the credentials provided to your organization by Aspera to access:

   https://downloads.asperasoft.com/en/downloads/2

   If you need help determining your firm's access credentials, contact your Aspera account manager.
2. For product upgrades, ensure you have prepared your system to upgrade to a newer version.

   Although the installer performs your upgrade automatically, Aspera *highly recommends* completing the tasks described in . If you do not follow these steps, you risk installation errors or losing your configuration settings.
3. Run the installer

   Run the following commands with the admin permissions. Replace the product version with that of your package.

OS	Commands

RedHat, zLinux, CentOS	`$ rpm -Uvh /path_to_installer/ aspera-desktopclient-version.rpm`
	
	Note: If your Linux OS is a minimal clean system, ensure that all the required dependencies are installed with your Aspera application by installing the product with a yum install:
	
	`$ yum --nogpgcheck install /path_to_installer/ aspera-desktopclient-version.rpm`

Configuring the Firewall

Desktop Client requires access through specific ports. If you cannot establish the connection, review your local corporate firewall settings and remove the port restrictions accordingly.

The following is basic information for configuring your firewall to allow Aspera file transfers. The outbound TCP port for SSH may differ depending on your organization's unique network settings. Although TCP/33001 is the default setting, refer to your IT Department for questions related to which SSH port(s) are open for file transfer. Consult your operating system's documentation for instructions on configuring your firewall. If your client host is behind a firewall that does not allow outbound connections, you will need to allow the following ports:

- **Outbound TCP/33001:** Allow outbound connections from the Aspera client on the TCP port (TCP/33001 by default, when connecting to a Windows server, or on another non-default port for other server operating systems).
- **Outbound UDP/33001:** Allow outbound connections from the Aspera client on the FASP UDP port (33001, by default).
- **Local firewall:** If you have a local firewall on the client (such as `iptables`), verify that it is not blocking your SSH and FASP transfer ports (such as TCP/UDP 33001).

Testing a Transfer

To make sure the software is working properly, set up a connection with the Aspera demo server and test downloads and uploads.

1. Download test files from the demo server.

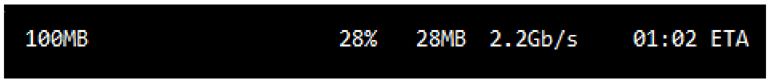
   Use the following command to download, press `y` to accept the server's key, and enter the password `demoaspera` when prompted:

   ```
   # ascp -T aspera@demo.asperasoft.com:aspera-test-dir-large/100MB /tmp/
   ```

   The transfer command is based on the following settings:

Item	Value
demo server address	`demo.asperasoft.com`
Login account	`aspera`
password	`demoaspera`
Test file	`/aspera-test-dir-large/100MB`
Download location	`/tmp/`
Transfer settings	`Fair` transfer policy, target rate 10M, minimum rate 1M, encryption disabled.

   You should see a message similar to the following:

   ```
   100MB                    28%   28MB  2.2Gb/s    01:02 ETA
   ```

   This message provides the following information:

Item	Description
100 MB	The name of the file that is being transferred.

Item	Description
28%	The percentage completed.
28 MB	The amount transferred.
2.2 Gbps	The current transfer rate.
01:02 ETA	The estimated time the transfer will complete.

2. Upload test files to the demo server.

   Run the command to upload the same file (`100MB`) back to the demo server, to its `/Upload` directory. Enter the password `demoaspera` when prompted:

```
# ascp -T /tmp/100MB aspera@demo.asperasoft.com:Upload/
```

Uninstalling

Desktop Client can be uninstalled without removing existing configuration files.

1. Close or stop the following applications and services:
   - FASP transfers
   - SSH connections
2. Uninstall Desktop Client by running the following command:

Platform	Command
RedHat, CentOS	`# rpm -e aspera-scp-client`
Debian	`# dpkg -P aspera-scp-client`

Note: This process does not remove Aspera configuration files. If you reinstall an Aspera product, these configuration files are applied to the new installation.

ascp: Transferring from the Command Line with Ascp

Ascp is a scriptable FASP transfer binary that enables you to transfer to and from Aspera transfer servers to which you have authentication credentials. Transfer settings are customizable and can include file manipulation on the source or destination, filtering of the source content, and client-side encryption-at-rest.

Ascp Command Reference

The `ascp` executable is a command-line FASP transfer program. This reference describes `ascp` syntax, command options, and supported environment variables.

For examples of `ascp` commands, see the following topics:

- Ascp General Examples on page 24
- Ascp File Manipulation Examples on page 26
- Ascp Transfers with Object Storage and HDFS on page 28

Another command-line FASP transfer program, Ascp 4 (`ascp4`), is optimized for transfers of many small files. It has many of the same capabilities as `ascp`, as well as its own features. For more information, see Introduction to Ascp 4 on page 53 and Comparison of Ascp and Ascp 4 Options on page 47.

Ascp Syntax

```
ascp options [[username@]src_host:]source1[ source2 ...]
 [[username@]dest_host:]dest_path
```

username

The username of the Aspera transfer user can be specified as part of the source or destination, whichever is the remote server. It can also be specified with the `--user` option. If you do not specify a username for the transfer, the local username is authenticated by default.

Note: If you are authenticating on a Windows computer as a domain user, the transfer server strips the domain from the username. For example, `Administrator` is authenticated rather than `DOMAIN\Administrator`. For this reason, you must specify the domain explicitly.

src_host

The name or IP address of the computer where the files or directories to be transferred reside.

source

The file or directory to be transferred. Separate multiple arguments with spaces.

dest_host

The name or IP address of the computer where the source files or directories are to be transferred.

dest_path

The destination directory where the source files or directories are to be transferred.

- If the source is a single file, the destination can be a filename. However, if there are multiple source arguments, the destination must be a directory.
- To transfer to the transfer user's docroot, specify "`.`" as the destination.
- If the destination is a symbolic link, then the file or directory is written to the target of the symbolic link.

Specifying Files, Directories, and Paths

- Specify paths on the remote computer relative to the transfer user's docroot. If the user has a restriction instead of a docroot, specify the full path, which must be allowed by the restriction.
- Avoid the following characters in file and directory names: `/ \ " : ' ? > < & * |`
- Specify paths with forward-slashes, regardless of the operating system.
- If directory or file arguments contain special characters, specify arguments with single-quotes (' ') to avoid interpretation by the shell.

URI paths: URI paths are supported, but with the following restrictions:

- If the source paths are URIs, they must all be in the same cloud storage account. No docroot (download), local docroot (upload), or source prefix can be specified.
- If a destination path is a URI, no docroot (upload) or local docroot (download) can be specified.
- The special schemes `stdio://` and `stdio-tar://` are supported on the client side only. They cannot be used for specifying an upload destination or download source.
- If required, specify the URI passphrase as part of the URI or set it as an environment variable (`ASPERA_SRC_PASS` or `ASPERA_DST_PASS`, depending on the transfer direction).

UNC paths: If the server is Windows and the path on the server is a UNC path (a path that points to a shared directory or file on Windows), it can be specified in an `ascp` command using one of the following conventions:

- As an UNC path that uses backslashes (`\`): If the client side is a Windows computer, the UNC path can be used with no alteration. For example, `\\192.168.0.10\temp`. If the client is not a Windows computer, every backslash in the UNC path must be replaced with two backslashes. For example, `\\\\192.168.0.10\\temp`.
- As an UNC path that uses forward slashes (`/`): Replace each backslash in the UNC path with a forward slash. For example, if the UNC path is `\\192.168.0.10\temp`, change it to `//192.168.0.10/temp`. This format can be used with any client-side operating system.

Testing paths: To test `ascp` transfers, use a `faux://` argument in place of the source or target path to send random data without writing it to disk at the destination. For more information, see Testing and Optimizing Transfer Performance on page 62. For examples, see Ascp General Examples on page 24.

Required File Access and Permissions

- Sources (for downloads) or destinations (for uploads) on the server must be in the transfer user's docroot or match one of the transfer user's file restrictions, otherwise the transfer stops and returns an error.
- The transfer user must have sufficient permissions to the sources or destinations, for example write access for the destination directory, otherwise the transfer stops and returns a permissions error.
- The transfer user must have authorization to do the transfer (upload or download), otherwise the transfer stops and returns a "management authorization refused" error.
- Files that are open for write by another process on a Windows source or destination cannot be transferred and return a "sharing violation" error. On Unix-like operating systems, files that are open for write by another process are transferred without reporting an error, but may produce unexpected results depending on what data in the file is changed and when relative to the transfer.

Environment Variables

The following environment variables can be used with the `ascp` command. The total size for environment variables depends on your operating system and transfer session. Aspera recommends that each environment variable value should not exceed 4096 characters.

`ASPERA_DST_PASS=`*password*

The password to authenticate a URI destination.

`ASPERA_LOCAL_TOKEN=`*token*

A token that authenticates the user to the client (in place of SSH authentication).

Note: If the local token is a basic or bearer token, the access key settings for cipher and preserve_time are not respected and the server settings are used. To set the cipher and timestamp preservation options as a client, set them in the command line.

`ASPERA_PROXY_PASS=`*proxy_server_password*

The password for an Aspera Proxy server.

`ASPERA_SCP_COOKIE=`*cookie*

A cookie string that you want associated with transfers.

`ASPERA_SCP_DOCROOT=`*docroot*

The transfer user docroot. Equivalent to using `--apply-local-docroot` when a docroot is set in `aspera.conf`.

`ASPERA_SCP_FILEPASS=`*password*

The passphrase to be used to encrypt or decrypt files. For use with `--file-crypt`.

`ASPERA_SCP_KEY="-----BEGIN RSA PRIVATE KEY..."`

The transfer user private key. Use instead of the `-i` option.

`ASPERA_SCP_PASS=`*password*

The password for the transfer user.

`ASPERA_SCP_TOKEN=`*token*

The transfer user authorization token. Overridden by `-W`.

`ASPERA_SRC_PASS=`***password***

The password to authenticate to a URI source.

Ascp Options

`-6`

Enable IPv6 address support. When specifying an IPv6 numeric host for *src_host* or *dest_host*, write it in brackets. For example, `username@[2001:0:4137:9e50:201b:63d3:ba92:da]:/path` or `--host=[fe80::21b:21ff:fe1c:5072%eth1]`.

`-@` ***range_start*****:*****range_end***

Transfer only part of a file: *range_start* is the first byte to send, and *range_end* is the last. If either position is unspecified, the file's first and last bytes (respectively) are assumed. This option only works for downloads of a single file and does not support transfer resume.

`-A, --version`

Display version and license information.

`--apply-local-docroot`

Apply the local docroot that is set in `aspera.conf` for the transfer user. Use to avoid entering object storage access credentials in the command line. This option is equivalent to setting the environment variable `ASPERA_SCP_DOCROOT`.

`-C` ***nodeid*****:*****nodecount***

Enable multi-session transfers (also known as parallel transfers) on a multi-node/multi-core system. A node ID (*nodeid*) and count (*nodecount*) are required for each session. *nodeid* and *nodecount* can be 1-128, but *nodeid* must be less than or equal to *nodecount*, such as 1:2, 2:2. Each session must use a different UDP port specified with the `-O` option. Large files can be split across sessions, see `--multi-session-threshold`. For more information, see the IBM Aspera High-Speed Transfer Server Admin Guide: Configuring Multi-Session Transfers.

`-c` ***cipher***

Encrypt in-transit file data using the specified cipher. Aspera supports three sizes of AES cipher keys (128, 192, and 256 bits) and supports two encryption modes, cipher feedback mode (CFB) and Galois/counter mode (GCM). The GCM mode encrypts data faster and increases transfer speeds compared to the CFB mode, but the server must support and permit it.

Cipher rules

The encryption cipher that you are allowed to use depends on the server configuration and the version of the client and server:

- When you request a cipher key that is shorter than the cipher key that is configured on the server, the transfer is automatically upgraded to the server configuration. For example, when the server setting is AES-192 and you request AES-128, the server enforces AES-192.
- When the server requires GCM, you must use GCM (requires version 3.9.0 or newer) or the transfer fails.
- When you request GCM and the server is older than 3.8.1 or explicity requires CFB, the transfer fails.
- When the server setting is "any", you can use any encryption cipher. The only exception is when the server is 3.8.1 or older and does not support GCM mode; in this case, you cannot request GCM mode encryption.
- When the server setting is "none", you must use "none". Transfer requests that specify an encryption cipher are refused by the server.

Cipher Values

Value	**Description**	**Support**
`aes128` `aes192` `aes256`	Use the GCM or CFB encryption mode, depending on the server configuration and version (see cipher negotiation matrix).	All client and server versions.
`aes128cfb` `aes192cfb` `aes256cfb`	Use the CFB encryption mode.	Clients version 3.9.0 and newer, all server versions.
`aes128gcm` `aes192gcm` `aes256gcm`	Use the GCM encryption mode.	Clients and servers version 3.9.0 and newer.
`none`	Do not encrypt data in transit. Aspera strongly recommends against using this setting.	All client and server versions.

Client-Server Cipher Negotiation

The following table shows which encryption mode is used depending on the server and client versions and settings:

	Server, v3.9.0+ **AES-XXX-GCM**	**Server, v3.9.0+** **AES-XXX-CFB**	**Server, v3.9.0+** **AES-XXX**	**Server, v3.8.1 or older** **AES-XXX**
Client, v3.9.0+ **AES-XXX-GCM**	GCM	server refuses transfer	GCM	server refuses transfer
Client, v3.9.0+ **AES-XXX-CFB**	server refuses transfer	CFB	CFB	CFB
Client, v3.9.0+ **AES-XXX**	GCM	CFB	CFB	CFB
Client, v3.8.1 or older **AES-XXX**	server refuses transfer	CFB	CFB	CFB

`--check-sshfp=`*fingerprint*

Compare *fingerprint* to the server SSH host key fingerprint that is set with `<ssh_host_key_fingerprint>` in `aspera.conf`. Aspera fingerprint convention is to use a hex string without the colons; for example, f74e5de9ed0d62feaf0616ed1e851133c42a0082. For more information on SSH host key fingerprints, see the IBM Aspera High-Speed Transfer Server Admin Guide: Securing your SSH Server.

Note: If HTTP fallback is enabled and the transfer "falls back" to HTTP, this option enforces server SSL certificate validation (HTTPS). Validation fails if the server has a self-signed certificate; a properly signed certificate is required.

`-D | -DD | -DDD`

Log at the specified debug level. With each `D`, an additional level of debugging information is written to the log.

`-d`

Create the destination directory if it does not already exist. This option is automatically applied to uploads to object storage.

`--delete-before-transfer`

Before transfer, delete any files that exist at the destination but not also at the source. The source and destination arguments must be directories that have matching names. Do not use with multiple sources, keepalive, URI storage, or HTTP fallback. The `asdelete` tool provides the same capability.

`--dest64`

Indicate that the destination path or URI is base64 encoded.

`-E` *'pattern'*

Exclude files or directories from the transfer based on matching the specified pattern to file names and paths (to exclude files by modification time, use `--exclude-newer-than` or `--exclude-older-than`). Use the `-N` option (include) to specify exceptions to `-E` rules. Rules are applied in the order in which they are encountered, from left to right. The following symbols can be used in the pattern:

- ***** (asterisk) represents zero or more characters in a string, for example `*.tmp` matches `.tmp` and `abcde.tmp`.
- **?** (question mark) represents a single character, for example `t?p` matches `tmp` but not `temp`.

For details and examples, see Using Filters to Include and Exclude Files on page 35.

Note: When filtering rules are found in `aspera.conf`, they are applied *before* rules given on the command line (`-E` and `-N`).

`-e` *prepost_script*

Run the specified pre-post script as an alternate to the default `aspera-prepost` script. Specify the full path to the pre-post script. Use pre-post scripts to run custom commands such as shell scripts, Perl scripts, Windows batch files, and executable binaries that can invoke a variety of environment variables. For instructions, see the IBM Aspera High-Speed Transfer Server Admin guide.

`--exclude-newer-than=`*mtime*, `--exclude-older-than=`*mtime*

Exclude files (but not directories) from the transfer, based on when the file was last modified. Positive *mtime* values are used to express time, in seconds, since the original system time (usually 1970-01-01 00:00:00). Negative *mtime* values (prefixed with "**-**") are used to express the number of seconds prior to the current time.

`-f` *config_file*

Read Aspera configuration settings from *config_file* rather than `aspera.conf`(the default).

`--file-checksum=`*hash*

Enable checksum reporting for transferred files, where *hash* is the type of checksum to calculate: `sha1`, `md5`, `sha-512`, `sha-384`, `sha-256`, or `none` (the default). When the value is `none`, the checksum that is configured on the server, if any, is used. For more information about checksum reporting, see *IBM Aspera High-Speed Transfer Server Admin Guide: Reporting Checksums*.

Important: When checksum reporting is enabled, transfers of very large files (>TB) take a long time to resume because the entire file must be reread.

`--file-crypt={encrypt|decrypt}`

Encrypt files (when sending) or decrypt files (when receiving) for client-side encryption-at-rest (EAR). Encrypted files have the file extension `.aspera-env`. This option requires the encryption/decryption passphrase to be set with the environment variable `ASPERA_SCP_FILEPASS`. If a client-side encrypted file is downloaded with an incorrect password, the download is successful, but the file remains encrypted and still has the file extension `.aspera-env`. For more information, see Client-Side Encryption-at-Rest (EAR) on page 46.

`--file-list=`*file*

Transfer all source files and directories listed in *file*. Each source item is specified on a separate line. UTF-8 file format is supported. Only the files and directories are transferred. Path information is not preserved at the destination. To read a file list from standard input, use "-" in place of *file*.

For example, if `list.txt` contains the following list of sources:

```
/tmp/code/compute.php
doc_dir
images/iris.png
images/rose.png
```

and the following command is run:

```
# ascp --file-list=list.txt --mode=send --user=username --
host=ip_addr .
```

then the destination, in this case the transfer user's docroot, will contain the following:

```
compute.php
doc_dir (and its contents)
iris.png
rose.png
```

Restrictions:

- The command line cannot use the *`user@host:source`* syntax. Instead, specify this information with the options `--mode`, `--host`, and `--user`.
- Paths specified in the file list cannot use the *`user@host:source`* syntax.
- Because multiple sources are being transferred, the destination must be a directory.
- Only one `--file-list` or `--file-pair-list` option is allowed per `ascp` session. If multiple lists are specified, only the last one is used.
- Only files and directories specified in the file list are transferred; any sources specified on the command line are ignored.
- If the source paths are URIs, the size of the file list cannot exceed 24 KB.

To create a file list that also specifies destination paths, use `--file-pair-list`.

`--file-manifest={none|text}`

Generate a list of all transferred files when set to `text`. Requires `--file-manifest-path` to specify the location of the list. (Default: `none`)

`--file-manifest-path=`*directory*

Save the file manifest to the specified location when using `--file-manifest=text`. File manifests must be stored locally. For cloud or other non-local storage, specify a *local* manifest path.

`--file-manifest-inprogress-suffix=`*suffix*

Apply the specified suffix to the file manifest's temporary file. For use with `--file-manifest=text`. (Default suffix: `.aspera-inprogress`)

`--file-pair-list=`*file*

Transfer files and directories listed in *file* to their corresponding destinations. Each source is specified on a separate line, with its destination on the line following it.

Specify destinations relative to the transfer user's docroot. Even if a destination is specified as an absolute path, the path at the destination is still relative to the docroot. Destination paths specified in the list are created automatically if they do not already exist.

For example, if the file `pairlist.txt` contains the following list of sources and destinations:

```
Dir1
Dir2
my_images/iris.png
project_images/iris.png
/tmp/code/compute.php
/tmp/code/compute.php
/tmp/tests/testfile
testfile2
```

and the following command is run:

```
# ascp --file-pair-list=pairlist.txt --mode=send --user=username
 --host=ip_addr .
```

then the destination, in this case the transfer user's docroot, now contains the following:

```
Dir2  (and its contents)
project_images/iris.png
tmp/code/compute.php
testfile2
```

Restrictions:

- The command line cannot use the *`user@host:source`* syntax. Instead, specify this information with the options `--mode`, `--host`, and `--user`.
- The *`user@host:source`* syntax cannot be used with paths specified in the file list.
- Because multiple sources are being transferred, the destination specified on the command line must be a directory.
- Only one `--file-pair-list` or `--file-list` option is allowed per `ascp` session. If multiple lists are specified, only the last one is used.
- Only files from the file pair list are transferred; any additional source files specified on the command line are ignored.
- If the source paths are URIs, the file list cannot exceed 24 KB.

For additional examples, see Ascp General Examples on page 24.

`-G` ***write_size***

If the transfer destination is a server, use the specified write-block size, which is the maximum number of bytes that the receiver can write to disk at a time. Default: 256 KB, Range: up to 500 MB. This option accepts suffixes "M" or "m" for *mega* and "K" or "k" for *kilo*, such that a *write_size* of `1M` is one MB.

This is a performance-tuning option that overrides the `write_block_size` set in the client's `aspera.conf`. However, the `-G` setting is overridden by the `write_block_size` set in the server's `aspera.conf`. The receiving server never uses the `write_block_size` set in the client's `aspera.conf`.

`-g` ***read_size***

If the transfer source is a server, use the specified read-block size, which is the maximum number of bytes that the sender reads from the source disk at a time. Default: 256 KB, Range: up to 500 MB. This option accepts suffixes "M" or "m" for *mega* and "K" or "k" for *kilo*, such that a *read_size* of `1M` is one MB.

This is a performance-tuning option that overrides the `read_block_size` set in the client's `aspera.conf`. However, the `-g` setting is overridden by the `read_block_size` set in the server's `aspera.conf`. When set to the default value, the read size is the default internal buffer size of the server, which might vary by operating system. The sending server never uses the `read_block_size` set in the client's `aspera.conf`.

`-h, --help`

Display the help text.

`--host=`*hostname*

Transfer to the specified host name or address. Requires `--mode`. This option can be used instead of specifying the host with the *hostname*:*file* syntax.

`-i` ***private_key_file***

Authenticate the transfer using public key authentication with the specified SSH private key file. The argument can be just the filename if the private key is located in *`user_home_dir`*`/.ssh/`, because `ascp` automatically searches for key files there. Multiple private key files can be specified by repeating the `-i` option. The keys are tried in order and the process ends when a key passes authentication or when all keys have been tried without success, at which point authentication fails.

`-K` ***probe_rate***

Measure bottleneck bandwidth at the specified probing rate (Kbps). (Default: 100Kbps)

`-k {0|1|2|3}`

Enable the resuming of partially transferred files at the specified resume level. (Default: 0)

Specify this option for the first transfer or it will not work for subsequent transfers. Resume levels:

- `-k 0` – Always re-transfer the entire file.
- `-k 1` – Compare file attributes and resume if they match, and re-transfer if they do not.
- `-k 2` – Compare file attributes and the sparse file checksums; resume if they match, and re-transfer if they do not.
- `-k 3` – Compare file attributes and the full file checksums; resume if they match, and re-transfer if they do not.

If a complete file exists at the destination (no `.aspx`), the source and destination file sizes are compared. If a partial file and a valid `.aspx` file exist at the destination, the source file size and the file size recorded in the `.aspx` file are compared.

Note: If the destination is a URI path, then the only valid options are `-k 0` and `-k 1` and no `.aspx` file is created.

`-L` ***local_log_dir*[:*size*]**

Log to the specified directory on the client computer rather than the default directory. Optionally, set the size of the log file (Default: 10 MB). See also `-R` for setting the log directory on the server.

`-l` ***max_rate***

Transfer at rates up to the specified target rate. (Default: 10000 Kbps) This option accepts suffixes "G" or "g" for *giga*, "M" or "m" for *mega*, "K" or "k" for *kilo*, and "P", "p", or "%" for percentage. Decimals are allowed. If this option is not set by the client, the setting in the server's `aspera.conf` is used. If a rate cap is set in the local or server `aspera.conf`, the rate does not exceed the cap.

`-m` ***min_rate***

Attempt to transfer no slower than the specified minimum transfer rate. (Default: 0) If this option is not set by the client, then the server's `aspera.conf` setting is used. If a rate cap is set in the local or server `aspera.conf`, then the rate does not exceed the cap.

`--mode={send|recv}`

Transfer in the specified direction: `send` or `recv` (receive). Requires `--host`.

`--move-after-transfer=`*archivedir*

Move source files and copy source directories to *archivedir* after they are successfully transferred. Because directories are copied, the original source tree remains in place. The transfer user must have write permissions to the *archivedir*. The *archivedir* is created if it does not already exist. If the archive directory cannot be created, the transfer proceeds and the source files remain in their original location.

To preserve portions of the file path above the transferred file or directory, use this option with `--src-base`. For an example, see Ascp File Manipulation Examples on page 26.

To remove empty source directories (except those specified as the source to transfer), use this option with `--remove-empty-directories` .

Restrictions:

- *archivedir* must be on the same file system as the source. If the specified archive is on a separate file system, it is created (if it does not exist), but an error is generated and files are not moved to it.
- For cloud storage, *archivedir* must be in the same cloud storage account as the source and must not already contain files with the same name (the existing files cannot be overwritten and the archiving fails).
- If the source is on a remote system (`ascp` is run in receive mode), *archivedir* is subject to the same docroot restrictions as the remote user.
- `--remove-after-transfer` and `--move-after-transfer` are mutually exclusive. Using both in the same session generates an error.
- Empty directories are not saved to *archivedir*.
- When used with `--remove-empty-directories` and `--src-base`, scanning for empty directories starts at the specified source base and proceeds down any subdirectories. If no source base is specified and a file path (as opposed to a directory path) is specified, then only the immediate parent directory is removed (if empty) after the source files have been moved.

`--multi-session-threshold=`*threshold*

Split files across multiple `ascp` sessions if their size is greater than or equal to *threshold*. Use with `-C`, which enables multi-session transfers.

Files whose sizes are less than *threshold* are not split. If *threshold* is set to 0 (the default), no files are split.

If `--multi-session-threshold` is not used, the threshold value is taken from the setting for `<multi_session_threshold_default>` in the `aspera.conf` file on the client. If not found in `aspera.conf` on the client, the setting is taken from `aspera.conf` on the server. The command-line setting overrides any `aspera.conf` settings, including when the command-line setting is 0 (zero).

Multi-session uploads to cloud storage are supported for S3 only and require additional configuration. For more information, see the IBM Aspera High-Speed Transfer Server Admin Guide: Configuring Multi-Session Transfers.

`-N` '*pattern*'

Include files or directories in the transfer based on matching the specified pattern to file names and paths. Rules are applied in the order in which they are encountered, from left to right, such that `-N` rules protect files from `-E` rules that follow them.

Note: An include rule **must** be followed by at least one exclude rule, otherwise all files are transferred because none are excluded. To exclude all files that do not match the include rule, use `-N '/**/' -E '/**'` at the end of your filter arguments.

The following symbols can be used in the pattern:

- ***** (asterisk) represents zero or more characters in a string, for example `*.tmp` matches `.tmp` and `abcde.tmp`.
- **?** (question mark) represents any single character, for example `t?p` matches `tmp` but not `temp`.

For details on specifying patterns and rules, including examples, see Using Filters to Include and Exclude Files on page 35.

Note: Filtering rules can also be specified in `aspera.conf`. Rules found in `aspera.conf` are applied *before* any `-E` and `-N` rules specified on the command line.

`-O` ***fasp_port***

Use the specified UDP port for FASP transfers. (Default: 33001)

`--overwrite={never|always|diff|diff+older|older}`

Overwrite destination files with source files of the same name. Default: `diff`. This option takes the following values:

- `never` - Never overwrite the file. However, if the parent folder is not empty, its access, modify, and change times may still be updated.
- `always` - Always overwrite the file.
- `diff` - Overwrite the file if different from the source. If a complete file at the destination is the same as a file on the source, it is not overwritten. Partial files are overwritten or resumed depending on the resume policy.
- `diff+older` - Overwrite the file if older and also different than the source. For example, if the destination file is the same as the source, but with a different timestamp, it will not be overwritten. Plus, if the destination file is different than the source, but newer, it will not be overwritten.
- `older` - Overwrite the file if its timestamp is older than the source timestamp.

Interaction with resume policy (-k): If the overwrite method is `diff` or `diff+older`, difference is determined by the resume policy (`-k {0|1|2|3}`). If `-k 0` or no `-k` is specified, the source and destination files are always considered different and the destination file is always overwritten. If `-k 1`, the source and destination files are compared based on file attributes (currently file size). If `-k 2`, the source and destination files are compared based on sparse checksums. If `-k 3`, the source and destination files are compared based on full checksums.

`-P` ***ssh-port***

Use the specified TCP port to initiate the FASP session. (Default: 22)

`-p`

Preserve file timestamps for access and modification time. Equivalent to setting `--preserve-modification-time` and `--preserve-access-time` (and `--preserve-creation-time` on Windows). Timestamp support in object storage varies by provider; consult your object storage documentation to determine which settings are supported.

On Windows, modification time may be affected when the system automatically adjusts for Daylight Savings Time (DST). For details, see the Microsoft KB article, http://support.microsoft.com/kb/129574.

On Isilon IQ OneFS systems, access time (`atime`) is disabled by default. In this case, `atime` is the same as `mtime`. To enable the preservation of `atime`, run the following command:

```
# sysctl efs.bam.atime_enabled=1
```

`--partial-file-suffix=`***suffix***

Enable the use of partial files for files that are in transit, and set the suffix to add to names of partial files. (The suffix does not include a "`.`", as for a file extension, unless explicitly specified as part of the suffix.) This option only takes effect when set on the receiver side. When the transfer is complete, the suffix is removed. (Default: suffix is null; use of partial files is disabled.)

`--policy={high|fair|low|fixed}`

Set the FASP transfer policy.

- `high` - Adjust the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, the transfer rate is twice as fast as a fair-policy transfer. The `high` policy requires maximum (target) and minimum transfer rates.
- `fair` - Adjust the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, bandwidth is shared fairly by transferring at an even rate. The `fair` policy requires maximum (target) and minimum transfer rates.
- `low` - Adjust the transfer rate to use the available bandwidth up to the maximum rate. Similar to fair mode, but less aggressive when sharing bandwidth with other network traffic. When congestion occurs, the transfer rate is reduced to the minimum rate until other traffic decreases.
- `fixed` - Attempt to transfer at the specified target rate, regardless of network or storage capacity. This can decrease transfer performance and cause problems on the target storage. Aspera discourages using the `fixed` policy except in specific contexts, such as bandwidth testing. The `fixed` policy requires a maximum (target) rate.

If `--policy` is not set, `ascp` uses the server-side policy setting (`fair` by default). If the server does not allow the selected policy, the transfer fails.

`--precalculate-job-size`

Calculate the total size before starting the transfer. The server-side `pre_calculate_job_size` setting in `aspera.conf` overrides this option.

`--preserve-access-time`

Preserve the source-file access timestamps at the destination. Because source access times are updated by the transfer operation, the timestamp preserved is the one just *prior* to the transfer. (To prevent access times at the source from being updated by the transfer operation, use the `--preserve-source-access-time` option.)

On Isilon IQ OneFS systems, access time (`atime`) is disabled by default. In this case, `atime` is the same as `mtime`. To enable the preservation of `atime`, run the following command:

```
# sysctl efs.bam.atime_enabled=1
```

`--preserve-acls={native|metafile|none}`

Preserve Access Control Lists (ACL) data for macOS, Windows, and AIX files. To preserve ACL data for other operating systems, use `--preserve-xattrs`. See also `--remote-preserve-acls`. Default: `none`.

- `native` - Preserve attributes using the native capabilities of the file system. This mode is only supported for Windows, macOS, and AIX. If the destination and source do not support the same native ACL format, `async` reports and error and exits.
- `metafile`- Preserve file attributes in a separate file, named `*filename*.aspera-meta`. For example, attributes for `readme.txt` are preserved in a second file named `readme.txt.aspera-meta`. These metafiles are platform independent and can be copied between hosts without loss of information. This mode is supported on all file systems.
- `none` - Do not preserve attributes. This mode is supported on all file systems.

Important Usage Information:

- ACLs are not preserved for directories.
- Both `--preserve-acls` and `--remote-preserve-acls` must be specified in order for the target side of a pull (Ascp with `--mode=recv`) to apply the ACLs.
- Very old versions of `ascp` do not support values other than `none`, and transfers using `native` or `metafile` fail with an error that reports incompatible FASP protocol versions.

`--preserve-creation-time`

(Windows only) Preserve source-file creation timestamps at the destination. Only Windows systems retain information about creation time. If the destination is not a Windows computer, this option is ignored.

`--preserve-file-owner-gid`, `--preserve-file-owner-uid`

(Linux, UNIX, and macOS only) Preserve the group information (`gid`) or owner information (`uid`) of the transferred files. These options require the transfer user to be authenticated as a superuser.

`--preserve-modification-time`

Set the modification time, the last time a file or directory was modified (written), of a transferred file to the modification of the source file or directory. Preserve source-file modification timestamps at the destination.

On Windows, modification time may be affected when the system automatically adjusts for Daylight Savings Time (DST). For details, see the Microsoft KB article, http://support.microsoft.com/kb/129574.

`--preserve-source-access-time`

Preserve the access times of the original sources to the last access times prior to transfer. This prevents access times at the source from being updated by the transfer operation. Typically used in conjunction with the `--preserve-access-time` option.

`--preserve-xattrs={native|metafile|none}`

Preserve extended file attributes data (xattr). Default: `none`. See also `--remote-preserve-xattrs`.

- `native` - Preserve attributes using the native capabilities of the file system. This mode is supported only on macOS and Linux. If the destination and source do not support the same native xattr format, `async` reports and error and exits. If the Linux user is not root, some attributes such as system group might not be preserved.
- `metafile`- Preserve file attributes in a separate file, named *`filename`*`.aspera-meta`. For example, attributes for `readme.txt` are preserved in a second file named `readme.txt.aspera-meta`. These metafiles are platform independent and can be copied between hosts without loss of information. This mode is supported on all file systems.
- `none` - Do not preserve attributes. This mode is supported on all file systems.

Important Usage Information:

- Extended attributes are not preserved for directories.
- If Ascp is run by a regular user, only user-level attributes are preserved. If run as superuser, all attributes are preserved.
- The amount of attribute data per file that can be transferred successfully is subject to `ascp`'s internal PDPU size limitation.
- Very old versions of Ascp do not support values other than `none`, and transfers using `native` or `metafile` fail with an error that reports incompatible FASP protocol versions.

`--proxy=`*proxy_url*

Use the proxy server at the specified address. *proxy_url* should be specified with the following syntax:

`dnat[s]://`*`proxy_username`*`:`*`proxy_password`*`@`*`server_ip_address`*`:`*`port`*

The default ports for DNAT and DNATS protocols are 9091 and 9092. For a usage example, see Ascp General Examples on page 24.

`-q`

Run `ascp` in quiet mode (disables the progress display).

`-R` ***remote_log_dir***

Log to the specified directory on the server rather than the default directory. **Note:** Client users restricted to aspshell are not allowed to use this option. To specify the location of the local log, use `-L`.

`--remote-preserve-acls={native|metafile|none}`

Like `--preserve-acls` but used when ACLs are stored in a different format on the remote computer. Defaults to the value of `--preserve-acls`.

Note: Both `--preserve-acls` and `--remote-preserve-acls` must be specified in order for the target side of a pull (Ascp with `--mode=recv`) to apply the ACLs.

`--remote-preserve-xattrs={native|metafile|none}`

Like `--preserve-xattrs` but used when attributes are stored in a different format on the remote computer. Defaults to the value of `--preserve-xattrs`.

`--remove-after-transfer`

Remove all source files, but not the source directories, once the transfer has completed successfully. Requires write permissions on the source.

`--remove-empty-directories`

Remove empty source directories once the transfer has completed successfully, but do not remove a directory specified as the source argument. To also remove the specified source directory, use `--remove-empty-source-directory`. Directories can be emptied using `--move-after-transfer` or `--remove-after-transfer`. Scanning for empty directories starts at the srcbase and proceeds down any subdirectories. If no source base is specified and a file path (as opposed to a directory path) is specified, then only the immediate parent directory is scanned and removed if it's empty following the move of the source file. **Note:** Do not use this option if multiple processes (`ascp` or other) might access the source directory at the same time.

`--remove-empty-source-directory`

Remove directories specified as the source arguments. For use with `--remove-empty-directories`.

`-S` ***remote_ascp***

Use the specified remote `ascp` binary, if different than `ascp`.

`--save-before-overwrite`

Save a copy of a file before it is overwritten by the transfer. A copy of `filename.ext` is saved as `filename.`*`yyyy.mm.dd.hh.mm.ss.index`*`.ext` in the same directory. *index* is set to 1 at the start of each second and incremented for each additional file saved during that second. The saved copies retain the attributes of the original. Not supported for URI path destinations.

`--skip-special-files`

Skip special files, such as devices and pipes, without reporting errors for them.

`--source-prefix=`*prefix*

Prepend *prefix* to each source path. The prefix can be a conventional path or a URI; however, URI paths can be used only if no docroot is defined.

`--source-prefix64=`*prefix*

Prepend the base64-encoded *prefix* to each source path. If `--source-prefix=`*prefix* is also used, the last option takes precedence.

`--src-base=`*prefix*

Strip the specified path prefix from the source path of each transferred file or directory. The remaining portion of the path remains intact at the destination.

Without `--src-base`, source files and directories are transferred without their source path. (However, directories do include their contents.)

Note: Sources located outside the source base are not transferred. No errors or warnings are issued, but the skipped files are logged.

Use with URIs: The `--src-base` option performs a character-to-character match with the source path. For object storage source paths, the prefix must specify the URI in the same manner as the source paths. For example, if a source path includes an embedded passphrase, the prefix must also include the embedded passphrase otherwise it will not match.

For examples, see Ascp File Manipulation Examples on page 26.

`--symbolic-links={follow|copy|copy+force|skip}`

Handle symbolic links using the specified method, as allowed by the server. For more information on symbolic link handling, see Symbolic Link Handling on page 41. On Windows, the only method is `skip`. On other operating systems, any of the following methods can be used:

- `follow` - Follow symbolic links and transfer the linked files. (Default)
- `copy` - Copy only the alias file. If a file with the same name is found at the destination, the symbolic link is not copied.
- `copy+force` - Copy only the alias file. If a file (not a directory) with the same name is found at the destination, the alias replaces the file. If the destination is a symbolic link to a directory, it's not replaced.
- `skip` - Skip symbolic links. Do not copy the link or the file it points to.

`-T`

Disable in-transit encryption for maximum throughput.

`--tags` ***string***

Metatags in JSON format. The value is limited to 4 Kb.

`--tags64` ***string***

Metatags in JSON format and base64 encoded. The value is limited to 4 Kb.

`-u` ***user_string***

Define a user string, such as variables, for pre- and post-processing. This string is passed to the pre- and -post-processing scripts as the environment variable `$USERSTR`.

`--user=`***username***

Authenticate the transfer using the specified username. Use this option instead of specifying the username as part of the destination path (as *user@host:file*).

Note: If you are authenticating on a Windows computer as a domain user, the transfer server strips the domain from the username. For example, `Administrator` is authenticated rather than `DOMAIN\Administrator`. For this reason, you must specify the domain explicitly.

`-v`

Run `ascp` in verbose mode. This option prints connection and authentication debug messages in the log file. For information on log files, see Log Files on page 64 .

`-W` **{*token_string*|@*token_file*}**

Authenticate using the authorization token string for the transfer, either as the string itself or when preceded with an `@`, the full path to the token file. This option takes precedence over the setting for the `ASPERA_SCP_TOKEN` environment variable.

`-wr`, `-wf`

Measure and report bandwidth from server to client (`-wr`) or client to server (`-wf`) before the transfer.

`-X` ***rexmsg_size***

Limit the size of retransmission requests to no larger than the specified size, in bytes. (Max: 1440)

`-Z` ***dgram_size***

Use the specified datagram size (MTU) for FASP transfers. Range: 296-65535 bytes. (Default: the detected path MTU)

As of version 3.3, datagram size can be specified on the server by setting `<datagram_size>` in `aspera.conf`. The server setting overrides the client setting, unless the client is using a version of `ascp` that is older than 3.3, in which case the client setting is used. If the pre-3.3 client does not set `-Z`, the datagram size is the discovered MTU and the server logs the message "LOG Peer client does not support alternative datagram size".

Ascp Options for HTTP Fallback

HTTP fallback serves as a secondary transfer method when the Internet connectivity required for Aspera FASP transfers (UDP port 33001, by default) is unavailable. When HTTP fallback is enabled and UDP connectivity is lost or cannot be established, the transfer will continue over the HTTP/S protocol.

Limitations:

- HTTP fallback must be enabled on the server.
- Folders that are symbolic links cannot be downloaded directly by using HTTP fallback. Folders that are symbolic links are processed correctly when their parent folder is the source.
- HTTP fallback can only follow symbolic links. Settings in `aspera.conf` or in the command line are ignored.
- HTTP fallback attempts to transfer at the target rate but is limited by TCP.
- HTTP fallback does not support pre-post processing or inline validation.

Options:

`-I` ***cert_file***

Certify fallback transfers with the specified HTTPS certificate file.

`-j {0|1}`

Encode all HTTP transfers as JPEG files when set to 1. (Default: 0)

`-t` ***port***

Transfer via the specified server port for HTTP fallback.

`-x` ***proxy_server***

Transfer to the specified proxy server address for HTTP fallback.

`-Y` ***key_file***

Certify HTTPS fallback transfers using the specified HTTPS transfer key.

`-y {0|1}`

If set to "`1`", use the HTTP fallback transfer server when a UDP connection fails. (Default: 0)

Ascp General Examples

Use the following Ascp examples to craft your own transfers.

To describe filepaths, use single-quote (' ') and forward-slashes (/) on all platforms. Avoid the following characters in filenames: `/ \ " : ' ? > < & * |`

- **Fair-policy transfer**

  Fair-policy transfer with maximum rate 100 Mbps and minimum at 1 Mbps, without encryption, transfer all files in `\local-dir\files` to 10.0.0.2:

  ```
  # ascp --policy=fair -l 100m -m 1m /local-dir/files root@10.0.0.2:/remote-dir
  ```

- **Fixed-policy transfer**

Fixed-policy transfer with target rate 100 Mbps, without encryption, transfer all files in `\local-dir\files` to 10.0.0.2:

```
# ascp -l 100m /local-dir/files root@10.0.0.2:/remote-dir
```

- **Specify UDP port for transfer**

  Transfer using UDP port 42000:

  ```
  # ascp -l 100m -O 42000 /local-dir/files user@10.0.0.2:/remote-dir
  ```

- **Public key authentication**

  Transfer with public key authentication using the key file `<home dir>/.ssh/aspera_user_1-key local-dir/files`:

  ```
  $ ascp -l 10m -i ~/.ssh/aspera_user_1-key local-dir/files root@10.0.0.2:/remote-dir
  ```

- **Username or filepath contains a space**

  Enclose the target in double-quotes when spaces are present in the username and remote path:

  ```
  # ascp -l 100m local-dir/files "User Name@10.0.0.2:/remote directory"
  ```

- **Content is specified in a file pair list**

  Specify source content to transfer to various destinations in a file pair list. Source content is specified using the full file or directory path. Destination directories are specified relative to the transfer user's docroot, which is specified as a "." at the end of the `ascp` command. For example, the following is a simple file pair list, `filepairlist.txt` that lists two source folders, `folder1` and `folder2`, with two destinations, `tmp1` and `tmp2`:

  ```
  /tmp/folder1
  tmp1
  /tmp/folder2
  tmp2
  ```

  ```
  # ascp --user=user_1 --host=10.0.0.2 --mode=send --file-pair-list=/tmp/
  filepairlist.txt .
  ```

  This command and file pair list create the following directories within the transfer user's docroot on the destination:

  ```
  /tmp1/folder1
  /tmp2/folder2
  ```

- **Network shared location transfer**

  Send files to a network shares location `\\1.2.3.4\nw-share-dir`, through the computer `10.0.0.2`:

  ```
  # ascp local-dir/files root@10.0.0.2:"//1.2.3.4/nw-share-dir/"
  ```

- **Parallel transfer on a multi-core system**

  Use parallel transfer on a dual-core system, together transferring at the rate 200Mbps, using UDP ports 33001 and 33002. Two commands are executed in different Terminal windows:

  ```
  # ascp -C 1:2 -O 33001 -l 100m /file root@10.0.0.2:/remote-dir &
  # ascp -C 2:2 -O 33002 -l 100m /file root@10.0.0.2:/remote-dir
  ```

- **Upload with content protection**

Upload the file `local-dir/file` to the server 10.0.0.2 with password protection (password: secRet):

```
# export ASPERA_SCP_FILEPASS=secRet ascp -l 10m --file-crypt=encrypt local-dir/file
 root@10.0.0.2:/remote-dir/
```

The file is saved on the server as `file.aspera-env`, with the extension indicating that the file is encrypted. See the next example for how to download and decrypt an encrypted file from the server.

- **Download with content protection and decryption**

  Download an encrypted file, `file.aspera-env`, from the server 10.0.0.2 and decrypt while transferring:

  ```
  # export ASPERA_SCP_FILEPASS=secRet; ascp -l 10m --file-crypt=decrypt root@10.0.0.2:/remote-
  dir/file.aspera-env  /local-dir
  ```

- **Decrypt a downloaded, encrypted file**

  If the password-protected file `file1` is downloaded on the local computer without decrypting, decrypt `file1.aspera-env` (the name of the downloaded/encrypted version of `file1`) to `file1`:

  ```
  $ export ASPERA_SCP_FILEPASS=secRet; /opt/aspera/bin/asunprotect -o file1 file1.aspera-env
  ```

- **Download through Aspera forward proxy with proxy authentication**

  User `Pat` transfers the file `/data/file1` to `/Pat_data/` on 10.0.0.2, through the proxy server at 10.0.0.7 with the proxy username `aspera_proxy` and password `pa33w0rd`. After running the command, Pat is prompted for the transfer user's (Pat's) password.

  ```
  # ascp --proxy dnats://aspera_proxy:pa33w0rd@10.0.0.7 /data/file1 Pat@10.0.0.2:/Pat_data/
  ```

Test transfers using `faux://`

For information on the syntax, see Testing and Optimizing Transfer Performance on page 62.

- **Transfer random data (no source storage required)**

  Transfer 20 GB of random data as user `root` to file `newfile` in the directory `/remote-dir` on 10.0.0.2:

  ```
  #ascp --mode=send --user=root --host=10.0.0.2 faux:///newfile?20g /remote-dir
  ```

- **Transfer a file but do not save results to disk (no destination storage required)**

  Transfer the file `/tmp/sample` as user `root` to 10.0.0.2, but do not save results to disk:

  ```
  #ascp --mode=send --user=root --host=10.0.0.2 /temp/sample faux://
  ```

- **Transfer random data and do not save result to disk (no source or destination storage required)**

  Transfer 10 MB of random data from 10.0.0.2 as user `root` and do not save result to disk:

  ```
  #ascp --mode=send --user=root --host=10.0.0.2 faux:///dummy?10m faux://
  ```

Ascp File Manipulation Examples

Ascp can manipulate files and directories as part of the transfer, such as upload only the files in the specified source directory but not the directory itself, create a destination directory, and move or delete source files after they are transferred.

- **Upload a directory**

  Upload the directory `/data/` to the server at 10.0.0.1, and place it in the `/storage/` directory on the server:

  ```
  # ascp /src/data/  root@10.0.0.1:/storage/
  ```

- **Upload only the contents of a directory (not the directory itself) by using the `--src-base` option:**

Upload only the contents of `/data/` to the `/storage/` directory at the destination. Strip the `/src/data/` portion of the source path and preserve the remainder of the file structure at the destination:

```
# ascp --src-base=/src/data/ /src/data/ root@10.0.0.1:/storage/
```

- **Upload a directory and its contents to a new directory by using the `-d` option**.

  Upload the `/data/` directory to the server and if it doesn't already exist, create the new folder `/storage2/` to contain it, resulting in `/storage2/data/` at the destination.

  ```
  # ascp -d /src/data/ root@10.0.0.1:/storage2/
  ```

- **Upload the contents of a directory, but not the directory itself, by using the `--src-base` option**:

  Upload all folders and files in the `/clips/out/` folder, but not the `out/` folder itself, to the `/in/` folder at the destination.

  ```
  # ascp -d --src-base=/clips/out/ /clips/out/ root@10.0.0.1:/in/
  ```

  Result: The source folders and their content appear in the `in` directory at the destination:

Source	**Destination**	**Destination without --src-base**
`/clips/out/file1`	`/in/file1`	`/in/out/file1`
`/clips/out/folderA/file2`	`/in/folderA/file2`	`/in/out/folderA/file2`
`/clips/out/folderB/file3`	`/in/folderB/file3`	`/in/out/folderB/file3`

  Without `--src-base`, the example command transfers not only the contents of the `out/` folder, but the folder itself.

  Note: Sources located outside the source base are not transferred. No errors or warnings are issued, but the skipped files are logged. For example, if `/clips/file4` were included in the above example sources, it would not be transferred because it is located outside the specified source base, `/clips/out/`.

- **Upload only the contents of a file and a directory to a new directory by using `--src-base`**

  Upload a file, `/monday/file1`, and a directory, `/tuesday/*`, to the `/storage/` directory on the server, while stripping the srcbase path and preserving the rest of the file structure. The content is saved as `/storage/monday/file1`and `/storage/tuesday/*` on the server.

  ```
  # ascp --src-base=/data/content /data/content/monday/file1 /data/content/
  tuesday/ root@10.0.0.1:/storage
  ```

- **Download only the contents of a file and a directory to a new directory by using `--src-base`**

  Download a file, `/monday/file1`, and a directory, `/tuesday/*`, from the server, while stripping the srcbase path and preserving the rest of the file structure. The content is saved as `/data/monday/file1` and `/data/tuesday/*` on the client.

  ```
  # ascp --src-base=/storage/content root@10.0.0.1:/storage/content/monday/
  file1 root@10.0.0.1:/storage/content/tuesday/ /data
  ```

- **Move the source file on the client after it is uploaded to the server by using `--move-after-transfer`**

  Upload`file0012` to Pat's docroot on the server at 10.0.0.1, and move (not copy) the file from `C:/Users/Pat/srcdir/` to `C:/Users/Pat/Archive` on the client.

  ```
  # ascp --move-after-transfer=C:/Users/Pat/Archive C:/Users/Pat/srcdir/
  file0012 Pat@10.0.0.1:/
  ```

- **Move the source file on the server after it is downloaded to the client by using `--move-after-transfer`**

Download `srcdir` from the server to `C:/Users/Pat` on the client, and move (not copy) `srcdir` to the archive directory `/Archive` on the server.

```
# ascp --move-after-transfer=Archive Pat@10.0.0.1:/srcdir C:/Users/Pat
```

- **Move the source file on the client after it is uploaded to the server and preserve the file structure one level above it by using `--src-base` and `--move-after-transfer`**

  Upload `file0012` to Pat's docroot on the server at 10.0.0.1, and save it as `/srcdir/file0012` (stripped of `C:/Users/Pat`). Also move `file0012` from `C:/Users/Pat/srcdir/` to `C:/Users/Pat/Archive` on the client, where it is saved as `C:/Users/Pat/Archive/srcdir/file0012`.

  ```
  # ascp --src-base=C:/Users/Pat --move-after-transfer=C:/Users/Pat/Archive
   C:/Users/Pat/srcdir/file0012 Pat@10.0.0.1:/
  ```

- **Delete a local directory once it is uploaded to the remote server by using `--remove-after-transfer` and `--remove-empty-directories`**

  Upload `/content/` to the server, then delete its contents (excluding partial files) and any empty directories on the client.

  ```
  # ascp -k2 -E "*.partial" --remove-after-transfer --remove-empty-
  directories /data/content root@10.0.0.1:/storage
  ```

- **Delete a local directory once its contents have been transferred to the remote server by using `--src-base`, `--remove-after-transfer`, and `--remove-empty-directories`**

  Upload `/content/` to the server, while stripping the srcbase path and preserving the rest of the file structure. The content is saved as `/storage/*` on the server. On the client, the contents of `/content/`, including empty directories but excluding partial files, are deleted.

  ```
  # ascp -k2 -E "*.partial" --src-base=/data/content --remove-after-transfer
   --remove-empty-directories /data/content root@10.0.0.1:/storage
  ```

Ascp Transfers with Object Storage and HDFS

Ascp transfers to and from servers in the cloud are similar to other Ascp transfers, though they might require explicit authorization to the storage as an authorization token or storage credentials.

Transfers with IBM Aspera On Demand and Cloud-Based HST Servers

Transfers to Aspera on Demand and cloud-based HST Servers require authorization credentials to the storage, but are otherwise the same as transfers to on-premises HST Server.

Provide object storage credentials in one of the following ways:

- Specify the storage password or secret key in the transfer user's docroot. (Preferred method)
- Set the storage password or secret key as an environment variable.
- Specify the storage password or secret key in the command line.

With Docroot Configured: Authenticate in the Docroot

If your transfer user account has a docroot set that includes credentials or credentials are configured in the `.properties` file, `ascp` transfers to and from Alibaba Cloud, Amazon S3, IBM COS - S3, Google Cloud Storage, Akamai, SoftLayer, Azure, and are the same as regular `ascp` transfers.

For instructions on configuring a docroot for these types of storage, see IBM Aspera High-Speed Transfer Server Admin Guide (Linux): Docroot Path Formatting for Cloud, Object, and HDFS Storage.

For command syntax examples, see Ascp General Examples on page 24. You are prompted for the transfer user's password when you run an `ascp` command unless you set the `ASPERA_SCP_PASS` environment variable or use SSH key authorization.

With No Docroot Configured: Authenticate with Environment Variables

Note: The `ASPERA_DEST_PASS` variable is not applicable to Google Cloud Storage or Amazon S3 using IAM roles.

Set an environment variable (`ASPERA_DEST_PASS`) with the storage password or access key:

```
# export ASPERA_DEST_PASS = secret_key
```

With `ASPERA_DEST_PASS` and `ASPERA_SCP_PASS` set, run `ascp` with the syntax listed in the table for transfers with no docroot configured, except that you do not need to include the storage password or access key, and are not prompted for the Aspera password upon running `ascp`.

With No Docroot Configured: Authenticate in the Command Line

If you do not have a docroot configured and do not set an environment variable (described previously), authenticate in the command line. In the following examples, the storage password or secret key are included as part of the destination path. You are prompted for the transfer user's password upon running `ascp` unless you set the `ASPERA_SCP_PASS` environment variable or use SSH key authorization.

Storage Platform	ascp Syntax and Examples
Alibaba Cloud	Aspera recommends running `ascp` transfers with Alibaba Cloud with a docroot configured.
Amazon S3	• If you are using IAM roles, you do not need to specify the access ID or secret key for your S3 storage. Upload syntax: `# ascp options --mode=send --user=username --` `host=s3_server_addr source_files s3://access_id:secret_key@s3.amazonaws` Upload example: `# ascp --mode=send --user=bear --` `host=s3.asperasoft.com bigfile.txt` `s3://1K3C18FBWF9902:GEyU...AqXuxtTVHWtc@s3.amazonaws.com/` `demos2014` Download syntax: `# ascp options --mode=recv --user=username --` `host=s3_server_addr s3://access_id:secret_key@s3.amazonaws.com/my_bucke` `my_source_path destination_path` Download example: `# ascp --mode=recv --user=bear --host=s3.asperasoft.com` `s3://1K3C18FBWF9902:GEyU...AqXuxtTVHWtc@s3.amazonaws.com/` `demos2014/bigfile.txt /tmp/`
Azure	These examples are for Azure blob storage. For Azure Files, use the syntax: `azure-files://storage_account:storage_access_key@file.core.windows.net/share`. Aspera recommends running `ascp` transfers with Azure Data Lake Storage with a docroot configured.

Storage Platform	ascp Syntax and Examples
	Upload syntax: `# ascp options --mode=send --user=username --` `host=server_address source_files azu://storage_account:storage_access_k` Upload example: `# ascp --mode=send --user=AS037d8eda429737d6 --` `host=dev920350144d2.azure.asperaondemand.com bigfile.txt` `azu://astransfer:zNfMtU...nBTkhB@blob.core.windows.net/abc` Download syntax: `# ascp options --mode=recv --user=username --` `host=server azu://storage_account:storage_access_key@blob.core.windows.` Download example: `# ascp --mode=recv --user=AS037d8eda429737d6 --` `host=dev920350144d2.azure.asperaondemand.com azu://` `astransfer:zNfMtU...nBTkhB@blob.core.windows.net/abc /` `downloads`
Google Cloud Storage	**Note:** The examples below require that the VMI running the Aspera server is a Google Compute instance. `# ascp options --mode=send --user=username --` `host=server_address source_files gs:///my_bucket/my_path` Upload example: `# ascp --mode=send --user=bear --host=10.0.0.5 bigfile.txt` `gs:///2017_transfers/data` Download syntax: `# ascp options --mode=recv --user=username --` `host=server gs:///my_bucket/my_path/source_file destination_path` Download example: `# ascp --mode=recv --user=bear --host=10.0.0.5` `gs:///2017_transfers/data/bigfile.txt /data`
HDFS	Aspera recommends running `ascp` transfers with HDFS with a docroot configured.
IBM COS - S3	Upload syntax: `# ascp options --mode=send --user=username --` `host=server_address source_files s3://access_id:secret_key@accessor_end`

Storage Platform	ascp Syntax and Examples

	Upload example:
	`# ascp --mode=send --user=bear --host=s3.asperasoft.com bigfile.txt s3://3ITI3OIUFEH233:KrcEW...AIuwQ@38.123.76.24/demo2017`
	Download syntax:
	`# ascp options --mode=send --user=username --host=server_address s3://access_id:secret_key@accessor_endpoint/vault_n source_files destination_path`
	Download example:
	`# ascp --mode=send --user=bear --host=s3.asperasoft.com s3://3ITI3OIUFEH233:KrcEW...AIuwQ@38.123.76.24/demo2017 /tmp/`

Writing Custom Metadata for Objects in Object Storage

Files that are uploaded to metadata-compatible storage (S3, Google Cloud, and Azure) can have custom metadata written with them by using the `--tags` or `--tags64` option. The argument is a JSON payload that specifies the metadata and that is base64 encoded if it is used as an argument for `--tags64`.

Metadata Behavior

- All objects that are uploaded in a session have the same metadata.
- If an upload resumes, the metadata of the original transfer is used.
- Multi-session transfers must specify the same metadata.
- Metadata are not retrieved when downloading objects; use the REST API associated with the storage.
- Transfers to object storages that do not support metadata (such as HDFS and Azure Files) fail if metadata is specified.

Specifying Metadata in JSON

The JSON payload has the general syntax of key-value pairs in a "cloud-metadata" section:

```
{
    "aspera": {
      "cloud-metadata": [
          {"key1":"value1"},
          {"key2":"value2"},
          ...
      ] } }
```

Restrictions on key-value pairs:

- *key* cannot be `ctime`, `mtime`, or `atime`. These keys are reserved and the transfer fails if they are used.
- *key* might be case-sensitive, depending on the destination storage type.
- The key-value pair must be less than 1024 characters.

Sample Ascp Session with Metadata

```
# ascp --tags='{"aspera":{"cloud-metadata":[{"location":"skellig"}]}}'
 --mode=send --user=rey --host=s3.asperasoft.com sourcefile.mov s3://
s3.amazonaws.com/project
```

Using Standard I/O as the Source or Destination

Ascp can use standard input (stdin) as the source or standard output (stdout) as the destination for a transfer, usually managed by using the Aspera FASP Manager SDK. The syntax depends on the number of files in your transfer; for single files use `stdio://` and for multiple files use `stdio-tar://`. The transfer is authenticated using SSH or a transfer token.

Named Pipes

A named pipe can be specified as a stdio destination, with the syntax `stdio:///`*`path`* for single files, or `stdio-tar:///`*`path`* for multiple files, where *path* is the path of the named pipe. If a docroot is configured on the destination, then the transfer goes to the named pipe *`docroot`*`/`*`path`*.

Note: Do not use `stdio:///`*`path`* to transfer multiple files. The file data is asynchronously concatenated in the output stream and might be unusable. Use `stdio-tar:///`*`path`* instead, which demarcates multiple files with headers.

Single File Transfers

To upload data that is piped into stdin, set the source as `stdio:///?`*`fsize`*, where*fsize* is the number of bytes (as a decimal) that are received from stdin. The destination is set as the path and filename. The file modification time is set to the time at which the upload starts. Standard input must transfer the exact amount of data that is set by *fsize*. If more or less data is received by the server, an error is generated.

To download data and pipe it into stdout, set the destination as `stdio://`.

Restrictions:

- `stdio://` cannot be used for persistent sessions. Use `stdio-tar://` instead.
- Only `--overwrite=always` or `--overwrite=never` are supported with `stdio://`. The behavior of `--overwrite=diff` and `--overwrite=diff+older` is undefined.

Single-file Transfer Examples:

- Upload 1025 bytes of data from the client stdin to `/remote-dir` on the server at 10.0.0.2. Save the data as the file `newfile`. Transfer at 100 Mbps.

```
file_source | ascp -l 100m --mode=send --user=username --host=10.0.0.2
 stdio:///?1025 /remote-dir/newfile
```

- Download the file `remote_file` from the server at 10.0.0.2 to stdout on the client. Transfer at 100 Mbps.

```
ascp -l 100m --mode=recv --user=username --host=10.0.0.2 remote_file
 stdio://
```

- Upload the file `local_file` to the server at 10.0.0.2 to the named pipe `/tmp/outpipe`. Transfer at 100 Mbps.

```
ascp -l 100m --mode=send --user=username --host=10.0.0.2 local_file
 stdio:////tmp/outpipe
```

Multi-File Transfers

Ascp can transfer one or more files in an encoded, streamed interface, similar to single file transfers. The primary difference is that the stream includes headers that demarcate data from individual files.

To upload files that are piped into stdin, set the source as `stdio-tar://`. The file modification time is set to the time at which the upload starts.

The file(s) in the input stream must be encoded in the following format. `File` can be the file name or file path, `Size` is the size of the file in bytes, and `Offset` is an optional parameter that sets where in the destination file to begin overwriting with the raw inline data:

```
[0 - n blank lines]
File: /path/to/file_1
Size: file_size
Offset: bytes

file_1 data
[0 - n blank lines]
File: /path/to/file_2
Size: file_size

file 2 data
...
```

To download one or more files to stdout, set the destination as `stdio-tar://`. Normal status output to stdout is suppressed during downloads because the transfer output is streamed to stdout. The data sent to stdout has the same encoding as described for uploads.

To download to a named pipe, set the destination to `stdio-tar:////path`, where *path* is the path of the named pipe.

When an offset is specified, the bytes that are sent replace the existing bytes in the destination file (if it exists). The bytes added to the destination file can extend beyond the current file size. If no offset is set, the bytes overwrite the file if overwrite conditions are met.

Restrictions:

- When downloading to `stdio-tar://`, the source list must consist of individual files only. Directories are not allowed.
- Only `--overwrite=always` or `--overwrite=never` are supported with `stdio-tar://`. The behavior of `--overwrite=diff` and `--overwrite=diff+older` is undefined.
- Offsets are only supported if the destination files are located in the native file system. Offsets are not supported for cloud destinations.

Multi-file Transfer Examples:

- Upload two files, `myfile1` (1025 bytes) and `myfile2` (20 bytes), to `/remote-dir` on the server at 10.0.0.2. Transfer at 100 Mbps.

  ```
  cat sourcefile | ascp -l 100m --mode=send --user=username --host=10.0.0.2
   stdio-tar:// /remote-dir
  ```

  Where `sourcefile` contains the following:

  ```
  File: myfile1
  Size: 1025

  << 1025 bytes of data>>
  File: myfile2
  Size: 20

  <<20 bytes of data>>
  ```

- Uploading multiple files from stdin by using a persistent session is the same as a non-persistent session.
- Update bytes 10-19 in file `/remote-dir/myfile1` on the server at 10.0.0.2 at 100 Mbps.

  ```
  cat sourcefile | ascp -l 100m --mode=send --user=username --host=10.0.0.2
   stdio-tar:// /remote-dir
  ```

Where `sourcefile` contains the following:

```
File: myfile1
Size: 10
Offset: 10

<< 10 bytes of data>>
```

- Upload two files, `myfile1` and `myfile2`, to the named pipe `/tmp/mypipe` (streaming output) on the server at 10.0.0.2. Transfer at 100 Mbps.

  ```
  ascp -l 100m --mode=send --user=username --host=10.0.0.2 myfile1 myfile2
   stdio-tar:////tmp/mypipe
  ```

  This sends an encoded stream of `myfile1` and `myfile2` (with the format of `sourcefile` in the upload example) to the pipe `/tmp/mypipe`. If `/tmp/mypipe` does not exist, it is created.
- Download the files from the previous example from 10.0.0.2 to stdout. Transfer at 100 Mbps.

  ```
  ascp -l 100m --mode=recv --user=username --host=10.0.0.2 myfile1 myfile2
   stdio-tar://
  ```

  Standard output receives data identical to `sourcefile` in the upload example.
- Download `/tmp/myfile1` and `/tmp/myfile2` to stdout by using a persistent session. Start the persistent session, which listens on management port 12345:

  ```
  ascp -l 100m --mode=recv --keepalive -M 12345 --user=username --
  host=10.0.0.2 stdio-tar://
  ```

  Send the following in through management port 12345:

  ```
   FASPMGR 2
   Type: START
   Source: /tmp/myfile1
   Destination: mynewfile1

   FASPMGR 2
   Type: START
   Source: /tmp/myfile2
   Destination: mynewfile2

   FASPMGR 2
   Type: DONE
  ```

  The destination must be a filename; file paths are not supported.

  Standard out receives the transferred data with the following syntax:

  ```
  File: mynewfile1
  Size: file_size

  mynewfile1_data
  File: mynewfile2
  Size: file_size

  mynewfile2_data
  ```

- Upload two files, `myfile1` and `myfile2`, to named pipe `/tmp/mypipe` on the server at 10.0.0.2. Transfer at 100 Mbps.

  ```
  ascp -l 100m --mode=send --user=username --host=10.0.0.2 myfile1 myfile2
   stdio-tar:////tmp/mypipe
  ```

  If file`/tmp/mypipe` does not exist, it is created.
- Upload two files, `myfile1` (1025 bytes) and `myfile2` (20 bytes) from stdio and regenerate the stream on the destination to send out through the named pipe `/tmp/mypipe` on the server at 10.0.0.2. Transfer at 100 Mbps.

  ```
  cat sourcefile | ascp -l 100m --mode=send --user=username --host=10.0.0.2
   stdio-tar:// stdio-tar:////tmp/pipe
  ```

  Where `sourcefile` contains the following:

  ```
  File: myfile1
  Size: 1025

  << 1025 bytes of data>>
  File: myfile2
  Size: 20

  <<20 bytes of data>>
  ```

Using Filters to Include and Exclude Files

Filters refine the list of source files (or directories) to transfer by indicating which to skip or include based on name matching. When no filtering rules are specified by the client, Ascp transfers all source files in the transfer list; servers cannot filter client uploads or downloads.

Command Line Syntax

`-E` '*pattern*'	Exclude files or directories with names or paths that match *pattern*.
`-N` '*pattern*'	Include files or directories with names or paths that match *pattern*.

Where:

- *pattern* is a file or directory name, or a set of names expressed with UNIX *glob* patterns.
- Surround patterns that contain wildcards with single quotes to prevent filter patterns from being interpreted by the command shell. Patterns that do not contain wildcards can also be in single quotes.

Basic usage

- Filtering rules are applied to the transfer list in the order they appear on the command line. If filtering rules are configured in `aspera.conf`, they are applied before the rules on the command line.
- Filtering is a process of exclusion, and `-N` rules override `-E` rules that follow them. `-N` cannot add back files that are excluded by a preceding exclude rule.
- An include rule **must** be followed by at least one exclude rule, otherwise all files are transferred because none are excluded. To exclude all files that do not match the include rule, use `-N '/**/' -E '/**'` at the end of your filter arguments.
- Filtering operates only on the set of files and directories in the transfer list. An include rule (`-N`) cannot add files or directories that are not already part of the transfer list.

Example	Transfer Result
`-E` '*rule*'	Transfer all files and directories except those with names that match *rule*.
`-N` '*rule*'	Transfer all files and directories because none are excluded.

Example	Transfer Result
	To transfer only the files and directories with names that match *rule* use: `ascp -N 'rule' -N '/**/' -E '/**'`
`-N` '*rule1*' `-E` '*rule2*'	Transfer all files and directories with names that match *rule1*, as well as all other files and directories except those with names that match *rule2*.
`-E` '*rule1*' `-N` '*rule2*'	Transfer all files and directories except those with names that match *rule1*. All files and directories not already excluded by *rule1* are included because no additional exclude rule follows `-N 'rule2'`. To transfer only the files and directories with names that do not match *rule1* but do match *rule2* use: `ascp -E 'rule1' -N 'rule2' -N '/**/' -E '/**'`

Filtering Rule Application

Filtering order

Filtering rules are applied to the transfer list in the order they appear on the command line.

1. Ascp compares the first file (or directory) in the transfer list to the pattern of the first rule.
2. If the file matches the pattern, Ascp includes it (`-N`) or excludes it (`-E`) and the file is immune to any following rules.

   Note: When a directory is excluded, directories and files in it are also excluded and are not compared to any following rules. For example, with the command-line options `-E '/images/' -N '/images/icons/'`, the directory `/images/icons/` is not included or considered because `/images/` was already excluded.
3. If the file does not match, Ascp compares it with the next rule and repeats the process for each rule until a match is found or until all rules have been tried.
4. If the file never matches any exclude rules, it is included in the transfer.
5. The next file or directory in the transfer list is then compared to the filtering rules until all eligible files are evaluated.

Example

Consider the following command:

```
# ascp -N 'file2' -E 'file[0-9]' /images/icons/ user1@examplehost:/tmp
```

Where `/images/icons/` is the source.

If `/images/icons/` contains `file1`, `file2`, `and fileA`, the filtering rules are applied as follows:

1. `file1` is compared with the first rule (`-N 'file2'`) and does not match so filtering continues.
2. `file1` is compared with the second rule (`-E 'file[0-9]`) and matches, so it is excluded from the transfer.
3. `file2` is compared with the first rule and matches, so it is included in the transfer and filtering stops for `file2`.
4. `fileA` is compared with the first rule and does not match so filtering continues.
5. `fileA` is compared with the second rule and does not match. Because no rules exclude it, `fileA` is included in the transfer.

   Note: If the filtering rules ended with `-N '/**/' -E '/**'`, then `fileA` would be excluded because it was not "protected" by an include rule.

Rule Patterns

Rule patterns (globs) use standard globbing syntax that includes wildcards and special characters, as well as several Aspera extensions to the standard.

- **Character case:** Case always matters, even if the file system does not enforce such a distinction. For example, on Windows FAT or NTFS file systems and macOS HPFS+, a file system search for "DEBUG" returns files "Debug" and "debug". In contrast, Ascp filter rules use exact comparison, such that "debug" does not match "Debug". To match both, use "[Dd]ebug".
- **Partial matches:** With globs, unlike standard regular expressions, the entire filename or directory name must match the pattern. For example, the pattern `abc*f` matches `abcdef` but not `abcdefg`.

Standard Globbing: Wildcards and Special Characters

`/`	The only recognized path separator.
`\`	Quotes any character literally, including itself. `\` is exclusively a quoting operator, not a path separator.
`*`	Matches zero or more characters, except "`/`" or the `.` in "`/.`".
`?`	Matches any single character, except "`/`" or the `.` in "`/.`".
`[ ... ]`	Matches exactly one of a set of characters, except "`/`" or the `.` in "`/.`".
`[^... ]`	When `^` is the first character, matches exactly one character *not* in the set.
`[!... ]`	When `!` is the first character, matches exactly one character *not* in the set.
`[`*x*`-`*x*`]`	Matches exactly one of a range of characters.
`[:`*xxxxx*`:]`	For details about this type of wildcard, see any POSIX-standard guide to globbing.

Globbing Extensions: Wildcards and Special Characters

no `/` or `*` at end of pattern	Matches files only.
`/` at end of pattern	Matches directories only. With `-N`, no files under matched directories or their subdirectories are included in the transfer. All subdirectories are still included, although their files will not be included. However, with `-E`, excluding a directory also excludes all files and subdirectories under it.
`*` or `/**` at end of pattern	Matches both directories and files.
`/**`	Like `*` but also matches "`/`" and the `.` in "`/.`".
`/` at start of pattern	Must match the entire string from the root of the transfer set. (Note: The leading `/` does not refer to the system root or the docroot.)

Standard Globbing Examples

Wildcard	Example	Matches	Does Not Match
`/`	`abc/def/xyz`	`abc/def/xyz`	`abc/def`
`\`	`abc\?`	`abc?`	`abc\? abc/D abcD`
`*`	`abc*f`	`abcdef abc.f`	`abc/f abcefg`
`?`	`abc??`	`abcde abc.z`	`abcdef abc/d abc/.`
`[ ... ]`	`[abc]def`	`adef cdef`	`abcdef ade`
`[^... ]`	`[^abc]def`	`zdef .def 2def`	`bdef /def /.def`

Wildcard	Example	Matches	Does Not Match
`[!...]`	`[!abc]def`	`zdef .def 2def`	`cdef /def /.def`
`[:`*xxxxx*`:]`	`[[:lower:]]def`	`cdef ydef`	`Adef 2def .def`

Globbing Extension Examples

Wildcard	Example	Matches	Does Not Match
`/**`	`a/**/f`	`a/f a/.z/f a/d/e/f`	`a/d/f/ za/d/f`
`*` at end of rule	`abc*`	`abc/ abcfile`	
`/**` at end of rule	`abc/**`	`abc/.file abc/d/e/`	`abc/`
`/` at end of rule	`abc/*/`	`abc/dir`	`abc/file`
no `/` at end of rule	`abc`	`abc` (file)	`abc/`
`/` at start of rule	`/abc/def`	`/abc/def`	`xyz/abc/def`

Testing Your Filter Rules

If you plan to use filtering rules, it's best to test them first. An easy way to test filtering rules, or to learn how they work, is to set up source and destination directories and use `demo.asperasoft.com` as the Aspera server:

1. On your computer, create a set of directories and files (size can be small) that approximate a typical transfer file set. In the following example, the file set is in `/tmp/src`.
2. Upload the file set to the Aspera demo server (demo.asperasoft.com) with the following command:

   ```
   # ascp /tmp/src aspera@demo.asperasoft.com:Upload/
   ```

   Where the user is "aspera" and the target is the `Upload` directory. At the prompt, enter the password "demoaspera".
3. Create a destination directory on your computer, for example `/tmp/dest`.
4. Download your files from the demo server to `/tmp/dest` to test your filtering rules. For example:

   ```
   # ascp -N 'wxy/**' -E 'def' aspera@demo.asperasoft.com:Upload/src/ /tmp/dest
   ```
5. Compare the destination directory with the source to determine if the filter behaved as expected.

   ```
   $ diff -r dest/ src/
   ```

   The `diff` output shows the missing files and directories (those that were not transferred).

Example Filter Rules

The example rules below are based on running a command such as the following to download a directory `AAA` from `demo.asperasoft.com` to `/tmp/dest`:

```
# ascp rules aspera@demo.asperasoft.com:Upload/AAA /tmp/dest
```

The examples below use the following file set:

```
AAA/abc/def
AAA/abc/.def
AAA/abc/.wxy/def
AAA/abc/wxy/def
AAA/abc/wxy/.def
AAA/abc/wxy/tuv/def
```

```
AAA/abc/xyz/def/wxy
AAA/wxyfile
AAA/wxy/xyx/
AAA/wxy/xyxfile
```

Key for interpreting example results below:

`< xxx/yyy` = Excluded
`xxx/yyy` = Included
`zzz/` = directory name
`zzz` = filename

1. Transfer everything except files and directories starting with "`.`":

   ```
   -N '*' -E 'AAA/**'
   ```

   Results:

   ```
   AAA/abc/def
   AAA/abc/wxy/def
   AAA/abc/wxy/tuv/def
   AAA/abc/xyz/def/wxy
   AAA/wxyfile
   AAA/wxy/xyx/
   AAA/wxy/xyxfile
   < AAA/abc/.def
   < AAA/abc/.wxy/def
   < AAA/abc/wxy/.def
   ```
2. Exclude directories and files whose names start with `wxy`:

   ```
   -E 'wxy*'
   ```

   Results:

   ```
   AAA/abc/def
   AAA/abc/.def
   AAA/abc/.wxy/def
   AAA/abc/xyz/def/
   < AAA/abc/wxy/def
   < AAA/abc/wxy/.def
   < AAA/abc/wxy/tuv/def
   < AAA/abc/xyz/def/wxy
   < AAA/wxyfile
   < AAA/wxy/xyx/
   < AAA/wxy/xyxfile
   ```
3. Include directories and files that start with "`wxy`" if they fall directly under `AAA`:

   ```
   -N 'wxy*' -E 'AAA/**'
   ```

   Results:

   ```
   AAA/wxy/
   AAA/wxyfile
   < AAA/abc/def
   < AAA/abc/.def
   < AAA/abc/.wxy/def
   < AAA/abc/wxy/def
   < AAA/abc/wxy/.def
   < AAA/abc/wxy/tuv/def
   < AAA/abc/xyz/def/wxy
   ```

```
< AAA/wxy/xyx/
< AAA/wxy/xyxfile
```

4. Include directories and files at any level that start with `wxy`, but do not include dot-files, dot-directories, or any files under the `wxy` directories (unless they start with `wxy`). However, subdirectories under `wxy` will be included:

   ```
   -N '*/wxy*' -E 'AAA/**'
   ```

   Results:

   ```
   AAA/abc/wxy/tuv/
   AAA/abc/xyz/def/wxy
   AAA/wxyfile
   AAA/wxy/xyx/
   < AAA/abc/def
   < AAA/abc/.def
   < AAA/abc/.wxy/def
   < AAA/abc/wxy/def     *
   < AAA/abc/wxy/.def
   < AAA/abc/wxy/tuv/def
   < AAA/wxy/xyxfile
   ```

   * Even though `wxy` is included, `def` is excluded because it's a file.
5. Include `wxy` directories and files at any level, even those starting with "`.`":

   ```
   -N '*/wxy*' -N '*/wxy/**' -E 'AAA/**'
   ```

   Results:

   ```
   AAA/abc/wxy/def
   AAA/abc/wxy/.def
   AAA/abc/wxy/tuv/def
   AAA/abc/xyz/def/wxy
   AAA/wxyfile
   AAA/wxy/xyx/
   AAA/wxy/xyxfile
   < AAA/abc/def
   < AAA/abc/.def
   < AAA/abc/.wxy/def
   ```
6. Exclude directories and files starting with `wxy`, but only those found at a specific location in the tree:

   ```
   -E '/AAA/abc/wxy*'
   ```

   Results:

   ```
   AAA/abc/def
   AAA/abc/.def
   AAA/abc/.wxy/def
   AAA/abc/xyz/def/wxy
   AAA/wxyfile
   AAA/wxy/xyx/
   AAA/wxy/xyxfile
   < AAA/abc/wxy/def
   < AAA/abc/wxy/.def
   < AAA/abc/wxy/tuv/def
   ```
7. Include the `wxy` directory at a specific location, and include all its subdirectories and files, including those starting with "`.`":

   ```
   -N 'AAA/abc/wxy/**' -E 'AAA/**'
   ```

Results:

```
AAA/abc/wxy/def
AAA/abc/wxy/.def
AAA/abc/wxy/tuv/def
< AAA/abc/def
< AAA/abc/.def
< AAA/abc/.wxy/def
< AAA/abc/xyz/def/wxy
< AAA/wxyfile
< AAA/wxy/xyx/
< AAA/wxy/xyxfile
```

Symbolic Link Handling

When transferring files using FASP (`ascp`, `ascp4`, or `async`), you can configure how the server and client handle symbolic links.

Note: Symbolic links are not supported on Windows. Server settings are ignored on Windows servers. If the transfer destination is a Windows computer, the only supported option that the client can use is **skip**.

Symbolic Link Handling Options and their Behavior

- **Follow**: Follow a symbolic link and transfer the contents of the linked file or directory as long as the link target is in the user's docroot.
- **Follow_wide** (Server only): For downloads, follow a symbolic link and transfer the contents of the linked file or directory **even if the link target is outside of the user's docroot**. Use caution with this setting because it might allow transfer users to access sensitive files on the server.
- **Create** (Server only): If the client requests to copy symbolic links in an upload, create the symbolic links on the server.
- **None** (Server only): Prohibit clients from creating symbolic links on the server; with this setting clients can only request to follow or skip symbolic links.
- **Copy** (Client only): Copy only the symbolic link. If a file with the same name exists at the destination, **the symbolic link does not replace the file**.
- **Copy+force** (Client only): Copy only the symbolic link. If a file with the same name exists at the destination, **the symbolic link replaces the file**. If the file of the same name at the destination is a symbolic link to a directory, it is not replaced.

  Note: A4 and Sync do not support the copy+force option.
- **Skip** (Client only): Skip symbolic links. Neither the link nor the file to which it points are transferred.

Symbolic link handling depends on the server configuration, the client handling request, and the direction of transfer, as described in the following tables. Multiple values can be set on the server as a comma-delimited list, such as the default "follow,create". In this case, the options are logically ORed based on the client's handling request.

Send from Client to Server (Upload)

	Server setting = create, follow (default)	**Server setting = create**	**Server setting = follow**	**Server setting = follow_wide**	**Server setting = none**
Client setting = follow (default for ascp and ascp4)	Follow	Follow	Follow	Follow	Follow

	Server setting = create, follow (default)	**Server setting = create**	**Server setting = follow**	**Server setting = follow_wide**	**Server setting = none**
Client setting = copy (default for async)	Copy	Copy	Skip	Skip	Skip
Client setting = copy+force	Copy and replace any existing files.	Copy and replace any existing files.	Skip	Skip	Skip
Client setting = skip	Skip	Skip	Skip	Skip	Skip

Receive to Client from Server (Download)

	Server setting = create, follow (default)	**Server setting = create**	**Server setting = follow**	**Server setting = follow_wide**	**Server setting = none**
Client setting = follow (default for ascp and ascp4)	Follow	Skip	Follow	Follow even if the target is outside the user's docroot.	Skip
Client setting = copy (default for async)	Copy	Copy	Copy	Copy	Copy
Client setting = copy+force	Copy and replace any existing files.	Copy and replace any existing files.	Copy and replace any existing files.	Copy and replace any existing files.	Copy and replace any existing files.
Client setting = skip	Skip	Skip	Skip	Skip	Skip

Server and Client Configuration

Server Configuration

To set symbolic link handling globally or per user, run the appropriate command:

```
# asconfigurator -x "set_node_data;symbolic_links,value"
# asconfigurator -x "set_user_data;user_name,username;symbolic_links,value"
```

Client Configuration

To specify symbolic link handling on the command line (with `ascp`, `ascp4`, or `async`), use `--symbolic-links=option`.

Creating SSH Keys

Public key authentication (SSH Key) is a more secure alternative to password authentication that allows users to avoid entering or storing a password, or sending it over the network. Public key authentication uses the client computer to generate the key-pair (a public key and a private key). The public key is then provided to the remote computer's administrator to be installed on that machine.

To log in into other Aspera servers with public key authentication, you need to generate a key-pair for the selected user account, as follows:

1. Create a `.ssh` directory in your home directory if it does not already exist:

   ```
   $ mkdir /home/username/.ssh
   ```

   Go to the `.ssh` folder:

   ```
   $ cd /home/username/.ssh
   ```

2. Run `ssh-keygen` to generate an SSH key-pair.

   Run the following command in the `.ssh` folder to create a key pair. For *`key_type`*, specify either RSA (`rsa`) or ECDSA (`ecdsa`). At the prompt for the key-pair's filename, press ENTER to use the default name `id_rsa` or `id_ecdsa`, or enter a different name, such as your username. For a passphrase, either enter a password, or press return twice to leave it blank:

   ```
   # ssh-keygen -t key_type
   ```

   Note: When you run `ascp` in FIPS mode (`<fips_enabled>` is set to `true` in `aspera.conf`), and you use passphrase-protected SSH keys, you must either (1) use keys generated by running `ssh-keygen` in a FIPS-enabled system, or (2) convert existing keys to a FIPS-compatible format using a command such as the following:

   ```
   # openssl pkcs8 -topk8 -v2 aes128 -in id_rsa -out new-id_rsa
   ```

3. Retrieve the public key file.

   The key-pair is generated to your home directory's `.ssh` folder. For example, assuming you generated the key with the default name `id_rsa`:

   `/home/username/.ssh/id_rsa.pub`

   Provide the public key file (for example, `id_rsa.pub`) to your server administrator so that it can be set up for your server connection.

4. Start a transfer using public key authentication with the `ascp` command.

   To transfer files using public key authentication on the command line, use the option `-i` *private_key_file*. For example:

   ```
   $ ascp -T -l 10M -m 1M  -i ~/.ssh/id_rsa  myfile.txt  jane@10.0.0.2:/space
   ```

   In this example, you are connecting to the server (`10.0.0.2`, directory `/space`) with the user account `jane` and the private key `~/.ssh/id_rsa`.

Reporting Checksums

File checksums are useful for trouble-shooting file corruption, allowing you to determine at what point in the transfer file corruption occurred. Aspera servers can report source file checksums that are calculated on-the-fly during transfer and then sent from the source to the destination.

To support checksum reporting, the transfer must meet both of the following requirements:

- Both the server and client computers must be running HST Server (formerly Enterprise Server and Connect Server) or HST Endpoint (formerly Point-to-Point Client) version 3.4.2 or higher.
- The transfer must be encrypted. Encryption is enabled by default.

The user on the destination can calculate a checksum for the received file and compare it (manually or programmatically) to the checksum reported by the sender. The checksum reported by the source can be retrieved in the destination logs, a manifest file, in IBM Aspera Console, or as an environment variable. Instructions for comparing checksums follow the instructions for enabling checksum reporting.

Checksum reporting is disabled by default. Enable and configure checksum reporting on the server by using the following methods:

- Edit `aspera.conf` with `asconfigurator`.
- Set `ascp` command-line options (per-transfer configuration).

Command-line options override the settings in `aspera.conf`.

Important: When checksum reporting is enabled, transfers of very large files (>TB) take a long time to resume because the entire file must be reread.

Overview of Checksum Configuration Options

`asconfigurator` Option **`ascp` Option**	**Description**

`file_checksum`
 	
`--file-checksum=`*`type`*
	Enable checksum reporting and specify the type of checksum to calculate for transferred files.
	`any` - Allow the checksum format to be whichever format the client requests. (Default in `aspera.conf`)
	`md5` - Calculate and report an MD5 checksum.
	`sha1` - Calculate and report a SHA-1 checksum.
	`sha256` - Calculate and report a SHA-256 checksum.
	`sha384` - Calculate and report a SHA-384 checksum.
	`sha512` - Calculate and report a SHA-512 checksum.
	Note: The default value for the `ascp` option is `none`, in which case the reported checksum is the one configured on the server, if any.

`file_manifest`
 	
`--file_manifest=`*`output`*
	The file manifest is a file that contains a list of content that was transferred in a transfer session. The file name of the file manifest is automatically generated from the transfer session ID.
	When set to `none`, no file manifest is created. (Default)
	When set to `text`, a text file is generated that lists all files in each transfer session.

`file_manifest_path`
 	
`--file_manifest_path=`*`path`*
	The location where manifest files are written. The location can be an absolute path or a path relative to the transfer user's home directory. If no path is specified (default), the file is generated under the destination path at the receiver, and under the first source path at the sender.
	Note: File manifests can be stored only locally. Thus, if you are using S3 or other non-local storage, you must specify a local manifest path.

Enabling checksum reporting by editing `aspera.conf`

To enable checksum reporting, run the following command:

```
# asconfigurator -x "set_node_data;file_checksum,checksum"
```

To enable and configure the file manifest where checksum report data is stored, run the following commands:

```
# asconfigurator -x "set_node_data;file_manifest,text"
# asconfigurator -x "set_node_data;file_manifest_path,filepath"
```

These commands create lines in `aspera.conf` as shown in the following example, where checksum type is `md5`, file manifest is enabled, and the path is `/tmp`.

```
<file_system>
    ...
    <file_checksum>md5</file_checksum>
    <file_manifest>text</file_manifest>
    <file_manifest_path>/tmp</file_manifest_path>
    ...
</file_system>
```

Enabling checksum reporting in an ascp session

To enable checksum reporting on a per-transfer-session basis, run `ascp` with the `--file-checksum=`*`hash`* option, where *`hash`* is `sha1`, `md5`, `sha-512`, `sha-384`, `sha-256`, or `none` (the default).

Enable the manifest with `--file-manifest=`*output* where *output* is either `text` or `none`. Set the path to the manifest file with `--file-manifest-path=`*path*.

For example:

```
# ascp --file-checksum=md5 --file-manifest=text --file-manifest-path=/
tmp file aspera_user_1@189.0.202.39:/destination_path
```

Setting up a Pre/Post-processing Script

An alternative to enabling and configuring the file manifest to collect checksum reporting is to set up a pre/post-processing script to report the values.

The checksum of a transferred file is stored in the pre/post environment variable `FILE_CSUM`, which can be used in pre/post scripts to output file checksums. For example, the following script outputs the checksum to the file `/tmp/cksum.log`:

```
#!/bin/bash
if [ $TYPE == File ]; then
    if [ $STARTSTOP == Stop ]; then
        echo "The file is: $FILE" >> /tmp/cksum.log
        echo "The file checksum is: $FILE_CSUM" >> /tmp/cksum.log
        chmod 777 $FILE
    fi
fi
```

For information on pre- and post-processing scripts and environment variables, see .

Comparing Checksums

If you open a file that you downloaded with Aspera and find that it is corrupted, you can determine when the corruption occurred by comparing the checksum that is reported by Aspera to the checksums of the files on the destination and on the source.

1. Retrieve the checksum that was calculated by Aspera as the file was transferred.
    - If you specified a file manifest and file manifest path as part of an `ascp` transfer or pre/post processing script, the checksums are in that file in the specified location.

- If you specified a file manifest and file manifest path in the GUI or `aspera.conf`, the checksums are in a file that is named `aspera-transfer-transfer_id-manifest.txt` in the specified location.

2. Calculate the checksum of the corrupted file. This example uses the MD5 checksum method; replace MD5 with the appropriate checksum method if you use a different one.

```
# md5sum filepath
```

3. Compare the checksum reported by Aspera with the checksum that you calculated for the corrupted file.

   - If they do not match, then corruption occurred as the file was written to the destination. Download the file again and confirm that it is not corrupted. If it is corrupted, compare the checksums again. If they do not match, investigate the write process or attempt another download. If they match, continue to the next step.
   - If they match, then corruption might have occurred as the file was read from the source. Continue to the next step.

4. Calculate the checksums for the file on the source. These examples use the MD5 checksum method; replace MD5 with the appropriate checksum method if you use a different one.

   Windows:

```
> CertUtil -hashfile filepath MD5
```

   Mac OS X:

```
$ md5 filepath
```

   Linux and Linux on z Systems:

```
# md5sum filepath
```

   AIX:

```
# csum -h MD5 filepath
```

   Solaris:

```
# digest -a md5 -v filepath
```

5. Compare the checksum of the file on the source with the one reported by Aspera.

   - If they do not match, then corruption occurred when the file was read from the source. Download the file again and confirm that it is not corrupted on the destination. If it is corrupted, continue to the next step.
   - If they match, confirm that the source file is not corrupted. If the source file is corrupted, replace it with an uncorrupted one, if possible, and then download the file again.

Client-Side Encryption-at-Rest (EAR)

Aspera clients can set their transfers to encrypt content that they upload to a server while it is in transit and stored on the server, a process known as client-side encryption-at-rest (EAR). The client specifies an encryption password and the files are uploaded to the server with a `.aspera-env` extension. Anyone downloading these `.aspera-env` files must have the password to decrypt them, and decryption can occur as the files are downloaded or later once they are physically moved to a computer with no network connection.

Implementation Notes:

- Client-side and server-side EAR can be used simultaneously, in which case files are doubly encrypted on the server.
- Servers can require client-side encryption. In this case, transfers that do not use client-side EAR fail with the error message, "Error: Server aborted session: Server requires content protection."

- Client-side encryption-at-rest is supported only for `ascp` transfers, and is not supported for `ascp4` or `async` transfers.

Using Client-Side EAR

Client-side EAR can be set in the `ascp` command line.

First, set the encryption and decryption password as the environment variable `ASPERA_SCP_FILEPASS`:

```
# export ASPERA_SCP_FILEPASS=password
```

For uploads (`--mode=send`), use `--file-crypt=encrypt`. For downloads (`--mode=recv`), use `--file-crypt=decrypt`.

```
# ascp --mode=send --file-
crypt=encrypt source_file user@host:/remote_destination
# ascp --mode=recv --file-crypt=decrypt user@host:/source_path/file.aspera-
env local_destination
```

For more command line examples, see Ascp General Examples on page 24.

Note: When a transfer to HST Server falls back to HTTP or HTTPS, client-side EAR is no longer supported. If HTTP fallback occurs while uploading, then the files are NOT encrypted. If HTTP fallback occurs while downloading, then the files remain encrypted.

Encrypting and Decrypting Files Outside of a Transfer

For particularly sensitive content, do not store unecrypted content on any computer with network access. Use an external drive to physically move encrypted files between computers. Desktop Client include the `asprotect` and `asunprotect` command-line tools that can be used to encrypt and decrypt files.

- To encrypt a file before moving it to a computer with network access, run the following command:

  ```
  # export ASPERA_SCP_FILEPASS=password;/opt/aspera/bin/asprotect -
  o file1.aspera-env file1
  ```

- To download client-side-encrypted files without decrypting them immediately, run the transfer without decryption enabled (do not specify `--file-crypt=decrypt` on the `ascp` command line).
- To decrypt encrypted files once they are on a computer with no network access, run the following command:

  ```
  # export ASPERA_SCP_FILEPASS=password;/opt/aspera/bin/asunprotect -
  o file1 file1.aspera-env
  ```

Comparison of Ascp and Ascp 4 Options

Many command-line options are the same for Ascp and Ascp 4; however, some options are available for only one or the behavior of an option is different. The following table lists the options that are available only for Ascp or Ascp 4, and the options that are available with both. If the option behavior is different, the Ascp option has ** added to the end and the difference is described following the table.

Ascp	Ascp 4
`-6`	
`-@[`*range_low*`:`*range_high*`]`	
`-A, --version`	`-A, --version`
`--apply-local-docroot`	

Ascp	**Ascp 4**
`-C` *nodeid:nodecount*	
`-c` *cipher*	`-c` *cipher*
`--check-sshfp=`*fingerprint*	
	`--chunk-size=`*bytes*
	`--compare=`*method*
	`--compression=`*method*
	`--compression-hint=`*num*
`-D | -DD | -DDD`	
`-d`	
	`--delete-before`
`--delete-before-transfer`**	`--delete-before-transfer`**
`--dest64`	
`-E` *pattern*	`-E` *pattern*
`-e` *prepost_filepath*	
	`--exclude-newer-than=`*mtime*
	`--exclude-older-than=`*mtime*
`-f` *config_file*	
	`--faspmgr-io`
`--file-checksum=`*hash*	
`--file-crypt={encrypt|decrypt}`	
`--file-list=`*filepath***	`--file-list=`*filepath***
`--file-manifest={none|text}`	
`--file-manifest-path=`*directory*	
`--file-manifest-inprogress-suffix=`*suffix*	
`--file-pair-list=`*filepath*	
`-G` *write_size*	
`-g` *read_size*	
`-h, --help`	`-h, --help`
`-i` *private_key_file_path***	`-i` *private_key_file_path*
`-K` *probe_rate*	
`-k {0|1|2|3}`	`-k {0|1|2|3}`
`--keepalive`	`--keepalive`
`-l` *max_rate*	`-l` *max_rate*
`-L` *local_log_dir*[:*size*]	`-L` *local_log_dir*[:*size*]

Ascp	**Ascp 4**
`-m` *min_rate*	`-m` *min_rate*
	`--memory=`*bytes*
	`--meta-threads=`*num*
`--mode={send|recv}`	`--mode={send|recv}`
`--move-after-transfer=`*archivedir*	
`--multi-session-threshold=`*threshold*	
`-N` *pattern*	`-N` *pattern*
	`--no-open`
	`--no-read`
	`--no-write`
`-O` *fasp_port*	`-O` *fasp_port*
`--overwrite=`*`method`***	`--overwrite=`*`method`***
`-P` *ssh-port*	`-P` *ssh-port*
`-p`	`-p`
`--partial-file-suffix=`*suffix*	
`--policy={fixed|high|fair|low}`	`--policy={fixed|high|fair|low}`
`--precalculate-job-size`	
`--preserve-access-time`	
`--preserve-acls=`*mode*	
`--preserve-creation-time`	
`--preserve-file-owner-gid`	`--preserve-file-owner-gid`
`--preserve-file-owner-uid`	`--preserve-file-owner-uid`
`--preserve-modification-time`	
`--preserve-source-access-time`	
`--preserve-xattrs=`*mode*	
`--proxy=`*proxy_url*	
`-q`	`-q`
`-R` *remote_log_dir*	`-R` *remote_log_dir*
	`--read-threads=`*num*
	`--remote-memory=`*bytes*
`--remote-preserve-acls=`*mode*	
`--remote-preserve-xattrs=`*mode*	
`--remove-after-transfer`	
`--remove-empty-directories`	

Ascp	Ascp 4
`--remove-empty-source-directory`	
	`--resume` (similar to `-k`)
`--retry-timeout=`*secs*	
`-S` *remote_ascp*	
`--save-before-overwrite`	
	`--scan-threads=`*num*
`--source-prefix=`*prefix*	
`--source-prefix64=`*prefix*	
	`--sparse-file`
`--src-base=`*prefix*	`--src-base=`*prefix*
`--symbolic-links=`*method***	`--symbolic-links=`*method***
`-T`	`-T`
`-u` *user_string*	`-u` *user_string*
`--user=`*username*	`--user=`*username*
`-v`	
`-W` *token_string* | @*token_filepath*	
`-w{r|f}`	
`-X` *rexmsg_size*	`-X` *rexmsg_size*
`-Z` *dgram_size*	`-Z` *dgram_size*

Differences in Option Behavior

`--delete-before-transfer`

With `ascp4`, `--delete-before-transfer` can be used with URI storage. URI storage is not supported for this option in `ascp`.

`--file-list`

`ascp` automatically applies `-d` if the destination folder does not exist. With `ascp4`, you must specify `-d`, otherwise all the files in the file list are written to a single file.

`-i` (SSH key authentication)

With `ascp`, the argument for `-i` can be just the file name of the private key file and `ascp` automatically looks in the `.ssh` directory of the user's home directory. With `ascp4`, the full or relative path to the private key file must be specified.

`--overwrite=`*`method`*

The default overwrite method is "diff" for `ascp` and "always" for `ascp4`.

`--symbolic-links`

Both `ascp` and `ascp4` support follow, copy, and skip, but only `ascp` supports copy+force.

Ascp FAQs

Answers to some common questions about controlling transfer behavior, such as bandwidth usage, resuming files, and overwriting files.

1. **How do I control the transfer speed?**

   You can specify a transfer policy that determines how a FASP transfer utilizes the network resource, and you can specify target and minimum transfer rates where applicable. In an `ascp` command, use the following flags to specify transfer policies that are fixed, fair, high, or low:

Policy	Command template
`Fixed`	`--policy=fixed -l` *`target_rate`*
`Fair`	`--policy=fair -l` *`target_rate`* `-m` *`min_rate`*
`High`	`--policy=high -l` *`target_rate`* `-m` *`min_rate`*
`Low`	`--policy=low -l` *`target_rate`* `-m` *`min_rate`*

   The policies have the following characteristics:

   - `high` - Adjust the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, the transfer rate is twice as fast as a fair-policy transfer. The `high` policy requires maximum (target) and minimum transfer rates.
   - `fair` - Adjust the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, bandwidth is shared fairly by transferring at an even rate. The `fair` policy requires maximum (target) and minimum transfer rates.
   - `low` - Adjust the transfer rate to use the available bandwidth up to the maximum rate. Similar to fair mode, but less aggressive when sharing bandwidth with other network traffic. When congestion occurs, the transfer rate is reduced to the minimum rate until other traffic decreases.
   - `fixed` - Attempt to transfer at the specified target rate, regardless of network or storage capacity. This can decrease transfer performance and cause problems on the target storage. Aspera discourages using the `fixed` policy except in specific contexts, such as bandwidth testing. The `fixed` policy requires a maximum (target) rate.

2. **What transfer speed should I expect? How do I know if something is "wrong" with the speed?**

   Aspera's FASP transport has no theoretical throughput limit. Other than the network capacity, the transfer speed may be limited by rate settings and resources of the computers. To verify that your system's FASP transfer can fulfill the maximum bandwidth capacity, prepare a client computer to connect to a server, and test the maximum bandwidth.

   Note: This test typically occupies most of a network's bandwidth. Aspera recommends this test be performed on a dedicated file transfer line or during a time of low network activity.

   On the client computer, start a transfer with fixed bandwidth policy. Start with a lower transfer rate and gradually increase the transfer rate toward the network bandwidth (for example, 1 MB, 5 MB, 10 MB, and so on). Monitor the transfer rate; at its maximum, it should be slightly below your available bandwidth:

   ```
   $ ascp -l 1m source-file destination
   ```

   To improve the transfer speed, also consider upgrading the following hardware components:

Component	Description
Hard disk	The I/O throughput, the disk bus architecture (such as RAID, IDE, SCSI, ATA, and Fiber Channel).
Network I/O	The interface card, the internal bus of the computer.
CPU	Overall CPU performance affects the transfer, especially when encryption is enabled.

3. **How do I ensure that if the transfer is interrupted or fails to finish, it will resume without re-transferring the files?**

   Use the `-k` flag to enable resume, and specify a resume rule:

   `-k 0` – Always re-transfer the entire file.
   `-k 1` – Compare file attributes and resume if they match, and re-transfer if they do not.
   `-k 2` – Compare file attributes and the sparse file checksums; resume if they match, and re-transfer if they do not.
   `-k 3` – Compare file attributes and the full file checksums; resume if they match, and re-transfer if they do not.

   Corruption or deletion of the `.asp-meta` file associated with an incomplete transfer will often result in a permanently unusable destination file even if the file transfer resumed and successfully transferred.

4. **How does Aspera handle symbolic links?**

   The `ascp` command follows symbolic links by default. This can be changed using `--symbolic-links=`*`method`* with the following options:

   - `follow` - Follow symbolic links and transfer the linked files. (Default)
   - `copy` - Copy only the alias file. If a file with the same name is found at the destination, the symbolic link is not copied.
   - `copy+force` - Copy only the alias file. If a file (not a directory) with the same name is found at the destination, the alias replaces the file. If the destination is a symbolic link to a directory, it's not replaced.
   - `skip` - Skip symbolic links. Do not copy the link or the file it points to.

   Important: On Windows, the only option is `skip`.

   Symbolic link handling also depends on the server configuration and the transfer direction. For more information, see Symbolic Link Handling on page 41.

5. **What are my choices for overwriting files on the destination computer?**

   In `ascp`, you can specify the `--overwrite=`*`method`* rule with the following method options:

   - `never` - Never overwrite the file. However, if the parent folder is not empty, its access, modify, and change times may still be updated.
   - `always` - Always overwrite the file.
   - `diff` - Overwrite the file if different from the source. If a complete file at the destination is the same as a file on the source, it is not overwritten. Partial files are overwritten or resumed depending on the resume policy.
   - `diff+older` - Overwrite the file if older and also different than the source. For example, if the destination file is the same as the source, but with a different timestamp, it will not be overwritten. Plus, if the destination file is different than the source, but newer, it will not be overwritten.
   - `older` - Overwrite the file if its timestamp is older than the source timestamp.

   Interaction with resume policy (-k): If the overwrite method is `diff` or `diff+older`, difference is determined by the resume policy (`-k {0|1|2|3}`). If `-k 0` or no `-k` is specified, the source and destination files are always considered different and the destination file is always overwritten. If `-k 1`, the source and destination files are compared based on file attributes (currently file size). If `-k 2`, the source and destination files are compared based on sparse checksums. If `-k 3`, the source and destination files are compared based on full checksums.

ascp4: Transferring from the Command Line with Ascp 4

Ascp 4 is a FASP transfer binary similar to Ascp but it has different strengths as well as capabilities that are unavailable with Ascp.

Introduction to Ascp 4

Ascp 4 is a FASP transfer binary that is optimized for sending extremely large sets of individual files. The executable, `ascp4`, is similar to `ascp` and shares many of the same options and capabilities, in addition to data streaming capabilities.

Both Ascp 4 and Ascp are automatically installed with IBM Aspera High-Speed Transfer Server, IBM Aspera High-Speed Transfer Endpoint, and IBM Aspera Desktop Client.

Ascp 4 Command Reference

Supported environment variables, the general syntax, and command options for `ascp4` are described in the following sections. `ascp4` exits with a `0` on success or a `1` on error. The error code is logged in the `ascp4` log file.

Note: Not all `ascp` options are available with `ascp4`. For more information, see Comparison of Ascp and Ascp 4 Options on page 47. Additionally, `ascp4` transfers fail if the user's docroot is a symbolic link, whereas `ascp` supports symbolic link docroots.

ascp4 Syntax

```
ascp4 options [[user@]srcHost:]source_file1[,source_file2,...]
 [[user@]destHost:]dest_path
```

User

The username of the Aspera transfer user can be specified as part of the as part of the source or destination, whichever is the remote server or with the `--user` option. If you do not specify a username for the transfer, the local username is authenticated by default.

Note: If you are authenticating on a Windows machine as a domain user, the transfer server strips the domain from the username. For example, `Administrator` is authenticated rather than `DOMAIN\Administrator`. Thus, you must specify the domain explicitly.

Source and destination paths

- If there are multiple source arguments, then the target path must be a directory.
- To describe filepaths, use single quotes (' ') and forward slashes (/) on all platforms.
- To transfer to the transfer user's docroot, specify `"."` as the destination.
- Avoid the following characters in filenames: `/ \ " : ' ? > < & * |`.
- If the destination is a symbolic link, then the file is written to the target of the symbolic link. However, if the symbolic link path is a relative path to a file (not a directory) and a partial file name suffix is configured on the receiver, then the destination path is relative to the user's home directory. Files within directories that are sent to symbolic links that use relative paths are not affected.

URI paths: URI paths are supported, but only with the following restrictions:

- If the source paths are URIs, they must all be in the same cloud storage account. No docroot (download), local docroot (upload), or source prefix can be specified.
- If a destination path is a URI, no docroot (upload) or local docroot (download) can be specified.

- The special schemes `stdio://` and `stdio-tar://` are supported only on the client side. They cannot be used as an upload destination or download source.
- If required, specify the URI passphrase as part of the URI or set it as an environment variable (`ASPERA_SRC_PASS` or `ASPERA_DST_PASS`, depending on the direction of transfer).

UNC paths: If the server is Windows and the path on the server is a UNC path (a path that points to a shared directory or file on Windows operating systems) then it can be specified in an `ascp4` command using one of the following conventions:

1. UNC path that uses backslashes (\ )

   If the client side is a Windows machine, the UNC path can be used with no alteration. For example, `\\192.168.0.10\temp`. If the client is not a Windows machine, every backslash in the UNC path must be replaced with two backslashes. For example, `\\\\192.168.0.10\\temp`.
2. UNC path that uses forward slashes (/)

   Replace each backslash in the UNC path with a forward slash. For example, if the UNC path is `\\192.168.0.10\temp`, change it to `//192.168.0.10/temp`. This format can be used with any client-side operating system.

Required File Access and Permissions

- Sources (for downloads) or destinations (for uploads) on the server must be in the transfer user's docroot or match one of the transfer user's file restrictions, otherwise the transfer stops and returns an error.
- The transfer user must have sufficient permissions to the sources or destinations, for example write access for the destination directory, otherwise the transfer stops and returns a permissions error.
- The transfer user must have authorization to do the transfer (upload or download), otherwise the transfer stops and returns a "management authorization refused" error.
- Files that are open for write by another process on a Windows source or destination cannot be transferred and return a "sharing violation" error. On Unix-like operating systems, files that are open for write by another process are transferred without reporting an error, but may produce unexpected results depending on what data in the file is changed and when relative to the transfer.

Environment Variables

If needed, you can set the following environment variables for use with an `ascp4` session. The total size for environment variables depends on your operating system and transfer session. Aspera recommends that each environment variable value should not exceed 4096 characters.

`ASPERA_SCP_PASS=`*password*

The password that is used for SSH authentication of the transfer user.

`ASPERA_SCP_TOKEN=`*token*

Set the transfer user authorization token. Ascp 4 currently supports transfer tokens, which must be created by using `astokengen` with the `--full-paths` option. For more information, see "Transfer Token Generation (astokengen)" in the IBM Aspera High-Speed Transfer Server Admin Guide.

`ASPERA_SCP_COOKIE=`*cookie*

A cookie string that is passed to monitoring services.

`ASPERA_SRC_PASS=`*password*

The password that is used to authenticate to a URI source.

`ASPERA_DST_PASS=`*password*

Set the password that is used to authenticate to a URI destination.

Ascp 4 Options

`-A, --version`

Display version and license information.

`-c {aes128|aes192|aes256|none}`

Encrypt in-transit file data using the specified cipher. This option overrides the `<encryption_cipher>` setting in `aspera.conf`.

`--check-sshfp=`*fingerprint*

Compare *fingerprint* to the server SSH host key fingerprint that is set with `<ssh_host_key_fingerprint>` in `aspera.conf`. Aspera fingerprint convention is to use a hex string without the colons; for example, f74e5de9ed0d62feaf0616ed1e851133c42a0082. For more information on SSH host key fingerprints, see the IBM Aspera High-Speed Transfer Server Admin Guide: Securing your SSH Server.

`--chunk-size=`*bytes*

Perform storage read/write operations with the specified buffer size. Also use the buffer size as an internal transmission and compression block. Valid range: 4 KB - 128 MB. For transfers with object storage, use `--chunk-size=1048576` if chunk size is not configured on the server to ensure that the chunk size of `ascp4` and Trapd match.

`--compare={size|size+mtime|md5|md5-sparse|sha1|sha1-sparse}`*method*

When using `--overwrite` and `--resume`, compare files with the specified method. If the `--overwrite` method is `diff` or `diff+older`, the default `--compare` method is `size`.

`--compression={none|zlib|lz4}`

Compress file data inline. Default: `lz4`. If set to `zlib`, `--compression-hint` can be used to set the compression level.

`--compression-hint=`*num*

Compress file data to the specified level when `--compression` is set to an option that accepts compression level settings (currently only zlib). A lower value results in less, but faster, data compression (0 = no compression). A higher value results in greater, slower compression. Valid values are -1 to 9, where -1 is "balanced". Default: -1.

`-D | -DD | -DDD`

Log at the specified debug level. With each `D`, an additional level of debugging information is written to the log. This option is not supported if the transfer user is restricted to aspshell.

`--delete-before`, `--delete-before-transfer`

Before transfer, delete files that exist at the destination but not at the source. The source and destination arguments must be directories that have matching names. Objects on the destination that have the same name but different type or size as objects on the source are not deleted. Do not use with multiple sources or `--keepalive`.

`-E` *pattern*

Exclude files or directories from the transfer based on the specified pattern. Use the `-N` option (include) to specify exceptions to `-E` rules. Rules are applied in the order in which they are encountered, from left to right. The following symbols can be used in the pattern:

- **`*`** (asterisk) represents zero or more characters in a string, for example `*.tmp` matches `.tmp` and `abcde.tmp`.
- **`?`** (question mark) represents a single character, for example `t?p` matches `tmp` but not `temp`.

For details and examples, see Using Filters to Include and Exclude Files on page 35.

Note: When filtering rules are found in `aspera.conf`, they are applied *before* rules given on the command line (`-E` and `-N`).

`--exclude-newer-than=`*`mtime`*

`--exclude-older-than=`*`mtime`*

Exclude files (but not directories) from the transfer based on when the file was last changed. Positive *mtime* values are used to express time, in seconds, since the original system time (usually 1970-01-01 00:00:00). Negative *mtime* values (prefixed with "`-`") are used to express the number of seconds prior to the current time.

`--faspmgr-io`

Run `ascp4` in API mode using FASP manager I/O. `ascp4` reads FASPMGR4 commands from management and executes them. The FASPMGR4 commands are PUT/WRITE/STOP to open/write/close on a file on the server.

`--file-list=`*filepath*

Transfer the files and directories that are listed in *filepath*. Only the files and directories are transferred; path information is not preserved at the destination. Each source must be specified on a separate line, for example:

```
src
src2
...
srcN
```

To read a file list from standard input, use "-" in place of *filepath* (as `ascp4 --file-list=- ...`) . UTF-8 file format is supported. Use with `-d` if the destination folder does not exist.

Restrictions:

- Paths in file lists cannot use *`user@host:filepath`* syntax. You must use `--user` with `--file-list`.
- Only one `--file-list` option is allowed per `ascp4` session. If multiple file lists are specified, all but the last are ignored.
- Only files and directories from the file list are transferred, and any additional source files or directories specified on the command line are ignored.
- If more than one read thread is specified (default is 2) for a transfer that uses `--file-list`, the files in the file list must be unique. Duplicates can produce unexpected results on the destination.
- Because multiple sources are being transferred, the destination must be a directory.
- If the source paths are URIs, the size of the file list cannot exceed 24 KB.

For very large file lists (~100 MB+), use with `--memory` to increase available buffer space.

`-h, --help`

Display the usage summary.

`--host=`*host*

Transfer to the specified host name or address. Requires `--mode`. This option can be used instead of specifying the host as part of the filename (as *hostname*:*filepath*).

`-i` *private_key_file*

Authenticate the transfer using public key authentication with the specified SSH private key file (specified with a full or relative path). The private key file is typically in the directory `$HOME/.ssh/`. If multiple `-i` options are specified, only the last one is used.

`-k {0|1|2|3}`

Enable the resumption of partially transferred files at the specified resume level. Default: 0. This option must be specified for your first transfer or it does not work for subsequent transfers. Resume levels:

- `-k 0`: Always re-transfer the entire file (same as `--overwrite=always`).
- `-k 1`: Compare file modification time and size and resume if they match (same as `--overwrite=diff --compare=size --resume`).

- `-k 2`: Compare sparse checksum and resume if they match (same as `--overwrite=diff --compare=md5-sparse --resume`).
- `-k 3`: Compare full checksum and resume if they match (same as `--overwrite=diff --compare=md5 --resume`).

`--keepalive`

Enable `ascp4` to run in persistent mode. This option enables a persistent session that does not require that source content and its destination are specified at execution. Instead, the persistent session reads source and destination paths through mgmt commands. Requires `--mode` and `--host`.

`-L` ***local_log_dir*[:*size*]**

Log to the specified directory on the client machine rather than the default directory. Optionally, set the size of the log file (default 10 MB).

`-l` ***max_rate***

Transfer at rates up to the specified target rate. Default: 10 Mbps. This option accepts suffixes "G/g" for Giga, "M/m" for Mega, "K/k" for Kilo, and "P/p/%" for percentage. Decimals are allowed. If this option is not set by the client, the server target rate is used. If a rate cap is set in the local or server `aspera.conf`, then the rate does not exceed the cap.

`-m` ***min_rate***

Attempt to transfer no slower than the specified minimum transfer rate. Default: 0. If this option is not set by the client, then the server's `aspera.conf` setting is used. If a rate cap is set in the local or server `aspera.conf`, then the rate does not exceed the cap.

`--memory=`*bytes*

Allow the local `ascp4` process to use no more than the specified memory. Default: 256 MB. See also `--remote-memory`.

`--meta-threads=`*num*

Use the specified number of directory "creation" threads (receiver only). Default: 2.

`--mode={send|recv}`

Transfer in the specified direction: `send` or `recv` (receive). Requires `--host`.

`-N` ***pattern***

Protect ("include") files or directories from exclusion by any `-E` (exclude) options that follow it. Files and directories are specified using *pattern*. Each option-plus-pattern is a *rule*. Rules are applied in the order (left to right) in which they're encountered. Thus, `-N` rules protect files only from `-E` rules that follow them. Create patterns using standard globbing wildcards and special characters such as the following:

- ***** (asterisk) represents zero or more characters in a string, for example `*.tmp` matches `.tmp` and `abcde.tmp`.
- **?** (question mark) represents any single character, for example `t?p` matches `tmp` but not `temp`.

For details on specifying patterns and rules, including examples, see Using Filters to Include and Exclude Files on page 35.

Note: Filtering rules can also be specified in `aspera.conf`. Rules found in `aspera.conf` are applied *before* any `-E` and `-N` rules specified on the command line.

`--no-open`

In test mode, do not actually open or write the contents of destination files.

`--no-read`

In test mode, do not read the contents of source files.

`--no-write`

In test mode, do not write the contents of destination files.

`-O` ***fasp_port***

Use the specified UDP port for FASP transfers. Default: 33001.

`--overwrite={always|never|diff|diff+older|older}`

Overwrite files at the destination with source files of the same name based on the *method*. Default: `always`. Use with `--compare` and `--resume`. *method* can be the following:

- `always` – Always overwrite the file.
- `never` – Never overwrite the file. If the destination contains partial files that are older or the same as the source files and `--resume` is enabled, the partial files resume transfer. Partial files with checksums or sizes that differ from the source files are not overwritten.
- `diff` – Overwrite the file if it is different from the source, depending on the `compare` method (default is `size`). If the destination is object storage, `diff` has the same effect as `always`.

  If `resume` is not enabled, partial files are overwritten if they are different from the source, otherwise they are skipped. If `resume` is enabled, only partial files with different sizes or checksums from the source are overwritten; otherwise, files resume.
- `diff+older` – Overwrite the file if it is older and different from the source, depending on the `compare` method (default is `size`). If `resume` is not enabled, partial files are overwritten if they are older and different from the source, otherwise they are skipped. If `resume` is enabled, only partial files that are different and older than the source are overwritten, otherwise they are resumed.
- `older` – Overwrite the file if its timestamp is older than the source timestamp.

`-P` ***ssh-port***

Use the specified TCP port to initiate the FASP session. (Default: 22)

`-p`

Preserve file timestamps for access and modification time. Equivalent to setting `--preserve-modification-time`, `--preserve-access-time`, and `--preserve-creation-time`. Timestamp support in object storage varies by provider; consult your object storage documentation to determine which settings are supported.

On Windows, modification time may be affected when the system automatically adjusts for Daylight Savings Time (DST). For details, see the Microsoft KB article, http://support.microsoft.com/kb/129574.

On Isilon IQ OneFS systems, access time (`atime`) is disabled by default. In this case, `atime` is the same as `mtime`. To enable the preservation of `atime`, run the following command:

```
# sysctl efs.bam.atime_enabled=1
```

`--policy={fixed|high|fair|low}`

Transfer according to the specified policy:

- `fixed` – Attempt to transfer at the specified target rate, regardless of network capacity. Content is transferred at a constant rate and the transfer finishes in a guaranteed time. The `fixed` policy can consume most of the network's bandwidth and is not recommended for most types of file transfers. This option requires a maximum (target) rate value (`-l`).
- `high` – Adjust the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, the transfer rate is twice as fast as transfer with a fair policy. This option requires maximum (target) and minimum transfer rates (`-l` and `-m`).
- `fair` – Adjust the transfer rate to fully utilize the available bandwidth up to the maximum rate. When congestion occurs, bandwidth is shared fairly by transferring at an even rate. This option requires maximum (target) and minimum transfer rates (`-l` and `-m`).
- `low` – Adjust the transfer rate to use the available bandwidth up to the maximum rate. Similar to fair mode, but less aggressive when sharing bandwidth with other network traffic. When congestion occurs, the transfer rate is reduced to the minimum rate until other traffic decreases.

If `--policy` is not set, `ascp4` uses the server-side policy setting (`fair` by default).

`--preserve-access-time`

Preserve the file timestamps (currently the same as `-p`).

`--preserve-creation-time`

Preserve the file timestamps (currently the same as `-p`).

`--preserve-file-owner-gid`

`--preserve-file-owner-uid`

(Linux, UNIX, and macOS only) Preserve the group information (`gid`) or owner information (`uid`) of the transferred files. These options require that the transfer user is authenticated as a superuser.

`--preserve-modification-time`

Preserve the file timestamps (currently the same as `-p`).

`--preserve-source-access-time`

Preserve the file timestamps (currently the same as `-p`).

`-q`

Run `ascp4` in quiet mode. This option disables the progress display.

`-R` ***remote_log_dir***

Log to the specified directory on the remote host rather than the default directory. **Note:** Client users that are restricted to aspshell are not allowed to use this option.

`--read-threads=`*num*

Use the specified number of storage "read" threads (sender only). Default: 2. To set "write" threads on the receiver, use `--write-threads`.

Note: If more than one read thread is specified for a transfer that uses `--file-list`, the files in the file list must be unique. Duplicates can produce unexpected results on the destination.

`--remote-memory=`*bytes*

Allow the remote `ascp4` process to use no more than the specified memory. Default: 256 MB.

`--resume`

Resume a transfer rather than re-transferring the content if partial files are present at the destination and they do not differ from the source file based on the `--compare` method. If the source and destination files do not match, then the source file is re-transferred. See `-k` for another way to enable resume.

`--scan-threads=`*num*

Use the specified number of directory "scan" threads (sender only). Default: 2.

`--sparse-file`

Enable `ascp4` to write sparse files to disk. This option prevents `ascp4` from writing zero content to disk for sparse files; `ascp4` writes a block to disk if even one bit is set in that block. If no bits are set in the block, `ascp4` does not write the block (`ascp4` blocks are 64 KB by default).

`--src-base=`*prefix*

Strip the specified prefix from each source path. The remaining portion of the source path is kept intact at the destination. Available only in send mode. For usage examples, see Ascp File Manipulation Examples on page 26.

Use with URIs: The `--src-base` option performs a character-to-character match with the source path. For object storage source paths, the prefix must specify the URI in the same manner as the source paths. For example, if a source path includes an embedded passphrase, the prefix must also include the embedded passphrase otherwise it will not match.

`--symbolic-links={follow|copy|skip}`

Handle symbolic links using the specified method. For more information on symbolic link handling, see Symbolic Link Handling on page 41. On Windows, the only option is `skip`. On other operating systems, this option takes following values:

- `follow` – Follow symbolic links and transfer the linked files. (Default)
- `copy` – Copy only the alias file. If a file with the same name exists on the destination, the symbolic link is not copied.
- `skip` – Skip symbolic links. Do not copy the link or the file it points to.

`-T`

Disable in-transit encryption for maximum throughput.

`-u` ***user_string***

Define a user string for pre- and post-processing. This string is passed to the pre- and -post-processing scripts as the environment variable `$USERSTR`.

`--user=`***username***

Authenticate the transfer using the specified username. Use this option instead of specifying the username as part of the destination path (as *user@host:file*).

Note: If you are authenticating on a Windows machine as a domain user, the transfer server strips the domain from the username. For example, `Administrator` is authenticated rather than `DOMAIN\Administrator`. Thus, you must specify the domain explicitly.

`--worker-threads=`***num***

Use the specified number of worker threads for deleting files. On the receiver, each thread deletes one file or directory at a time. On the sender, each thread checks for the presences of one file or directory at a time. Default: 1.

`--write-threads=`***num***

Use the specified number of storage "write" threads (receiver only). Default: 2. To set "read" threads on the sender, use `--read-threads`.

For transfers to object or HDFS storage, write threads cannot exceed the maximum number of jobs that are configured for Trapd. Default: 15. To use more threads, open `/opt/aspera/etc/trapd/trap.properties` on the server and set `aspera.session.upload.max-jobs` to a number larger than the number of write threads. For example,

```
# Number of jobs allowed to run in parallel for uploads.
# Default is 15
aspera.session.upload.max-jobs=50
```

`-X` ***rexmsg_size***

Limit the size of retransmission requests to no larger than the specified size, in bytes. Max: 1440.

`-Z` ***dgram_size***

Use the specified datagram size (MTU) for FASP transfers. Range: 296-65535 bytes. Default: the detected path MTU.

As of version 3.3, datagram size can be specified on the server by setting `<datagram_size>` in `aspera.conf`. The server setting overrides the client setting, unless the client is using a version of `ascp` that is older than 3.3, in which case the client setting is used. If the pre-3.3 client does not set `-Z`, the datagram size is the discovered MTU and the server logs the message "LOG Peer client doesn't support alternative datagram size".

Ascp 4 Transfers with Object Storage

Files that are transferred with object storage are sent in chunks through the Aspera Trapd service. By default, `ascp4` uses 64 KB chunks and Trapd uses 1 MB chunks; this mismatch in chunk size can cause `ascp4` transfers to fail.

To avoid this problem, take one of the following actions:

1. Set the chunk size (in bytes) in the server's `aspera.conf`. This value is used by both `ascp4` and Trapd, so the chunk sizes match.

   To set a global chunk size, run the following command:

   ```
   # asconfigurator -x
    "set_node_data;transfer_protocol_options_chunk_size,value"
   ```

   Where *value* is between 256 KB (262144 bytes) and 1 MB (1048576 bytes).

   To set a chunk size for the user, run the following command:

   ```
   # asconfigurator -x
    "set_user_data;user_name, username;transfer_protocol_options_chunk_size,value"
   ```
2. Set the chunk size in the client's `aspera.conf` to the Trapd chunk size.

   If Trapd is using the default chunk size, run the following command to set the chunk size to 1 MB:

   ```
   # asconfigurator -x
    "set_node_data;transfer_protocol_options_chunk_size,1048576"
   ```
3. Set the chunk size in the client command line.

   Run the `ascp4` session with the chunk size setting: `--chunk-size=1048576`.

Ascp 4 Examples

The command options for `ascp4` are generally similar to those for `ascp`. The following examples demonstrate options that are unique to Ascp 4. These options enable reading management commands and enable read/write concurrency.

For Ascp examples, see Ascp Command Reference on page 9 and Ascp Transfers with Object Storage and HDFS on page 28. See Comparison of Ascp and Ascp 4 Options on page 47 for differences in option availability and behavior.

- **Read FASP4 management commands**

  Read management commands V4 from management port 5000 and execute the management commands. The management commands version 4 are PUT, WRITE and CLOSE.

  ```
  # ascp4 -L /tmp/client-logs -R /tmp/server-logs --faspmgr-io -M 5000
   localhost:/tmp
  ```
- **Increase concurrency**

  The following command runs `ascp4` with two scan threads and eight read threads on the client, and eight meta threads and 16 write threads on the server.

  ```
  # ascp4 -L /tmp/logs -R /tmp/logs -l1g --scan-threads=2 --read-threads=8
   --write-threads=16 --meta-threads=8  /data/100K aspera@10.0.113.53:/data
  ```

Appendix

Restarting Aspera Services

When you change product settings, you might need to restart certain Aspera services in order for the new values to take effect.

IBM Aspera Central

If asperacentral is stopped, or if you have modified the `<central_server>` or `<database>` sections in `aspera.conf`, then you need to restart the service.

Run the following command in a Terminal window to restart asperacentral:

```
# systemctl restart asperacentral
```

or for Linux systems that use `init.d`:

```
# service asperacentral restart
```

IBM Aspera NodeD

Restart asperanoded if you have modified any setting in `aspera.conf`.

Run the following commands to restart asperanoded:

```
# systemctl restart asperanoded
```

or for Linux systems that use `init.d`:

```
# service asperanoded restart
```

Testing and Optimizing Transfer Performance

To verify that your system's FASP transfer is reaching the target rate and can use the maximum bandwidth capacity, prepare a client to connect to an Aspera server. For these tests, you can transfer an existing file or file set, or you can transfer uninitialized data in place of a source file, which you can destroy at the destination, eliminating the need to read from or write to disk and saving disk space.

Using faux:/// as a Test Source or Destination

You can use `faux:///` as the argument for the source or destination of an Ascp session to test data transfer without reading from disk on the source and writing to disk on the target. The argument takes different syntax depending on if you are using it as a mock source file or mock source directory.

Note: If you set very large file sizes (> PB) in a `faux:///` source, Aspera recommends that you use `faux://` as a target on the destination because most computers do not have enough system memory available to handle files of this size and your transfer might fail.

Faux Source File

To send random data in place of a source file (do not read from the source), you can specify the file as `faux:///`*`fname`*`?`*`fsize`*. *fname* is the name assigned to the file on the destination and *fsize* is the number of bytes to send. *fsize* can be set with modifiers (k/K, m/M, g/G, t/T, p/P, or e/E) to a maximum of 7x2^{60} bytes (7 EiB).

For example:

```
# ascp --mode=send --user=username --host=host_ip_address faux:///fname?fsize target_path
```

Faux Source Directory

In some cases, you might want to test the transfer of an entire directory, rather than a single file. Specify the faux source directory with the following syntax:

```
faux:///dirname?
file=file&count=count&size=size&inc=increment&seq=sequence&buf_init=buf_option
```

Where:

- *dirname* is a name for the directory (required)
- *file* is the root for file names, default is "file" (optional)
- *count* is the number of files in the directory (required)
- *size* is the size of the first file in the directory, default 0 (optional). *size* can be set with modifiers (k/K, m/M, g/G, t/T, p/P, or e/E) to a maximum of 7x2^{60} bytes (7 EiB).
- *increment* is the increment of bytes to use to determine the file size of the next file, default 0 (optional)
- *sequence* is how to determine the size of the next file: "sequential" or "random". Default is "sequential" (optional). When set to "sequential", file size is calculated as:

  ```
  size + ((N - 1) * increment)
  ```

  Where *N* is the file index; for the first file, *N* is one.

  When set to "random", file size is calculated as:

  ```
  size +/- (rand * increment)
  ```

  Where *rand* is a random number between zero and one. If necessary, *increment* is automatically adjusted to prevent the file size from being negative.

  For both options, *increment* is adjusted to prevent the file size from from exceeding 7x2^{60} bytes.
- *buf_option* is how faux source data are initialized: "none", "zero", or "random". Default is "zero". "none" is not allowed for downloads (Ascp run with `--mode=recv`).

When the defaults are used, Ascp sends a directory that is named *dirname* and that contains *count* number of zero-byte files that are named file_*count*.

For example, to transfer a faux directory ("mydir") that contains 1 million files to `/tmp` on 10.0.0.2, and the files in `mydir` are named "testfile" and file size increases sequentially from 0 to 2 MB by an increment of 2 bytes:

```
# ascp --mode=send --user=username --host=10.0.0.2 faux:///mydir?
file=testfile&count=1m&size=0&inc=2&seq=sequential /tmp
```

Faux Target

To send data but not save the results to disk at the destination (do not write to the target), specify the target as `faux://`.

For example, to send a real file to a faux target, run the following command:

```
# ascp --mode=send --user=username --host=host_ip_address source_file1 faux://
```

To send random data to a faux target, run the following command:

```
# ascp --mode=send --user=username --host=host_ip_address faux:///fname?fsize faux://
```

Testing Transfer Performance

1. Start a transfer with fair transfer policy and compare the transfer rate to the target rate.

   On the client computer, open the user interface and start a transfer (either from the GUI or command line). Click **Details** to open the Transfer Monitor.

   To leave more network resources for other high-priority traffic, use the **Fair** policy and adjust the target rate and minimum rate by sliding the arrows or entering values.
2. Test the maximum bandwidth.

   Note: This test will typically occupy a majority of the network's bandwidth. Aspera recommends performing it on a dedicated file transfer line or during a time of very low network activity.

   Use **Fixed** policy for the maximum transfer speed. Start with a lower transfer rate and increase gradually toward the network bandwidth.

Hardware Upgrades for Better Performance

To improve the transfer speed, you can also upgrade the related hardware components:

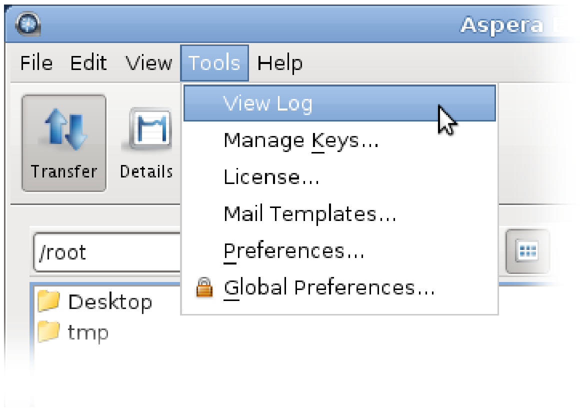
Component	**Description**
Hard disk	The I/O throughput, the disk bus architecture (such as RAID, IDE, SCSI, ATA, and Fiber Channel).
Network I/O	The interface card, the internal bus of the computer.
CPU	Overall CPU performance affects the transfer, especially when encryption is enabled.

Log Files

The application log file includes detailed transfer information and can be useful for review and support requests. You can configure log rotation and redirect Aspera logging so that it is not recorded in the system log file.

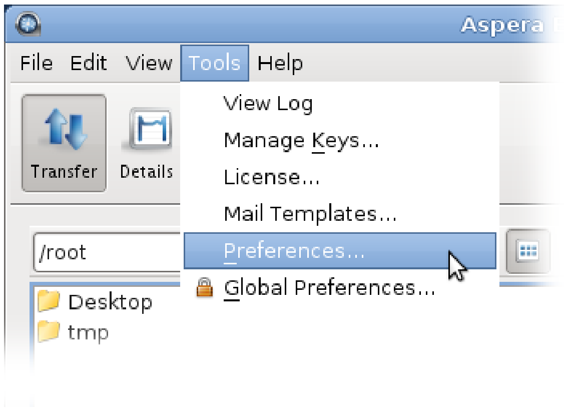
Viewing Logs and Setting Log Preferences

To view the log, from the GUI, click **Tools** > **View Log**.

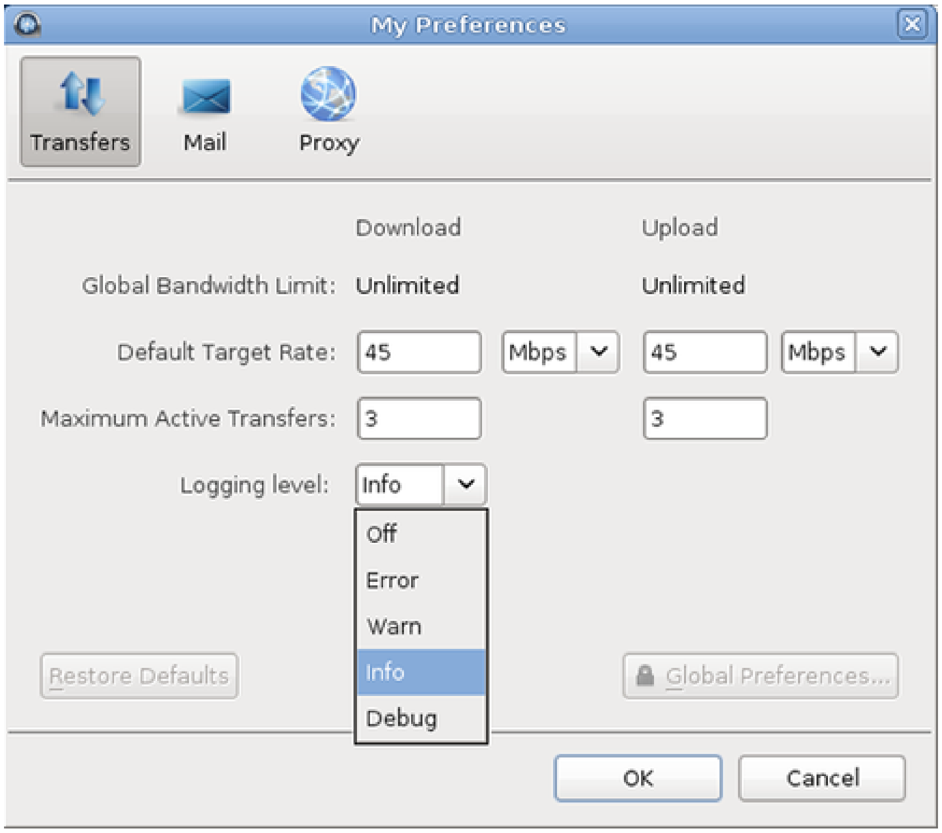


Note: To view logs from the command line in Linux, you must have a functional web-browser or other default application for opening HTML files.

To set the logging level for transfers, open the **My Preferences** dialog by clicking **Tools** > **Preferences** or by clicking **Preferences** in the upper-right corner of the application window.



The five logging levels to select from are: **Off**, **Error**, **Warn**, **Info**, and **Debug**. The system default is **Info**.



	Download		Upload	
Global Bandwidth Limit:	Unlimited		Unlimited	
Default Target Rate:	45	Mbps	45	Mbps
Maximum Active Transfers:	3		3	
Logging level:	Info			

Off
Error
Warn
Info
Debug

Redirecting Aspera Logging to a Different Location

On Linux systems, the application transfer logs are recorded in the system log file. Instead of mixing Aspera logging with system logging, you may want to redirect Aspera logging to a separate log file of your choice.

RedHat, CentOS, and Debian

On RedHat, CentOS, and Debian, the transfer logs are recorded in the following log file: `/var/log/messages`

To redirect Aspera logging, modify `/etc/syslog.conf` (`/etc/rsyslog.conf` in the case of Red Hat or CentOS 6.XA) and add `local2.none` to the `/var/log/messages` line. For example, if you have the following line:

```
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
```

Change it to:

```
*.info;mail.none;authpriv.none;cron.none;local2.none   /var/log/messages
```

Next, forward `local2.info` log messages to your new file. For example, to write to `/var/log/aspera.log`, add the following line just below the line you modified above:

```
local2.info        -/var/log/aspera.log
```

The log file name should be separated from the log facility (`local2.info`) by tab characters, not spaces and be preceded by a hyphen. The hyphen before the log file name allows for asynchronous logging.

Next, restart the syslog daemon to have it load the new configuration:

```
# service syslog restart
```

In the case of Red Hat or CentOS 6.X:

```
# service rsyslog restart
```

Your Aspera log messages now appear in `/var/log/aspera.log` instead of `/var/log/messages`.

SLES (Suse) systems

On SLES (Suse) systems, the transfer logs are recorded in the following system log file: `/var/log/localmessages`

To redirect Aspera logging, locate the following section in `/etc/syslog-ng/syslog-ng.conf`:

```
filter f_local { facility(local0, local1, local2, local3, local4, local5,
 local6, local7); };

destination localmessages { file("/var/log/localmessages"); };
log { source(src); filter(f_local); destination(localmessages); };
```

Modify the section as follows:

```
filter f_local { facility(local0, local1, local3, local4, local5, local6,
 local7); };
filter f_aspera { facility(local2); };

destination localmessages { file("/var/log/localmessages"); };
log { source(src); filter(f_local); destination(localmessages); };

destination asperalog { file("/var/log/aspera.log"); };
log { source(src); filter(f_aspera); destination(asperalog); };
```

Then run the following command:

```
# rcsyslog restart
```

Your Aspera log messages now appear in `/var/log/aspera.log` instead of `/var/log/localmessages`.

To test this, run the following commands:

```
# logger -p local2.info aspera test
# cat /var/log/aspera.log
```

The `cat` command should display something similar to the following:

```
Jun 13 10:30:33 linux-kua5 root: aspera test
```

Rotating Your Aspera Log File

There are several ways to rotate Aspera logs in Linux:

1. Add `/var/log/aspera.log` to `/etc/logrotate.d/syslog`.
2. Create an entry for `aspera.log` in `/etc/logrotate.conf`.
3. Create a separate configuration file for `aspera.log` in `/etc/logrotate.d/`.

The first option rotates your logs with the system logs (usually once a week, compressed, and saving the last 10 logs). On some servers, there is so much traffic that the logs need to be rotated more often than once a week, in which case option 2 or 3 should be used.

1. Add `/var/log/aspera.log` to the entries in `/etc/logrotate.d/syslog`, as follows:

```
/var/log/messages /var/log/secure /var/log/maillog /var/log/spooler /var/
log/boot.log /var/log/cron /var/log/aspera.log
{
    sharedscripts
    postrotate
    /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null ||
 true
    /bin/kill -HUP `cat /var/run/rsyslogd.pid 2> /dev/null` 2> /dev/null ||
 true
    endscript
}
```

2. Edit `/etc/logrotate.conf` by adding the configuration after the line "`# system-specific logs may also be configured here.`" The following example compresses and rotates 10 logs whenever `/var/log/aspera.log` reaches 100MB. After log rotation is complete, it runs whatever scripts are specified by `postrotate ... endscript`.

```
/var/log/aspera.log {
    rotate 10
    size 100M
    create 664 root
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
    compress
}
```

The following example compresses and rotates 10 logs once daily. Instead of moving the original log file and creating a new one, the `copytruncate` option tells `logrotate` to first copy the original log file, then truncate it to zero bytes.

```
/var/log/aspera.log {
    daily
    rotate 10
    copytruncate
    compress
}
```

3. Create a separate `/etc/logrotate.d/aspera` configuration file containing the same information as option 2.

Preserving IBM Spectrum Scale ACLs of Transferred Files

Ascp and Aspera Sync can preserve NFSv4 and POSIX ACLs and immutability attributes when transferring files from an IBM Spectrum Scale (formerly GPFS) cluster to another cluster.

Preserving Spectrum Scale ACLs

To preserve file attributes and permissions when transferring from one Spectrum Scale cluster to another, run `ascp` or `async` with the `--preserve-xattrs=native` option.

Preserving Expiration Attributes of Immutable Filesets

To preserve expiration attributes, use timestamp preservation options:

- **Ascp:** Use `--preserve-access-time` to preserve only the expiration attributes or `-p` to preserve all timestamps.
- **Aspera Sync:** Use`--preserve-access-time`.

Important Behavior Notes

Attribute preservation:

- Pools, replication, and cloning attributes are not preserved from source to destination.
- In Aspera Sync transfers, security attributes are preserved only when the user on both endpoints is root.

Immutable files and directories:

- Immutable directories on the source are not set as immutable on the destination. This ensures that the contents of the directory can be transferred.
- When the destination file already exists and the source file is changed to immutable (`immutable: yes`), Ascp and Aspera Sync update the destination file from mutable to immutable. However, if the source file is changed back to mutable (`immutable: no`), the change cannot be applied to the destination file because it is still immutable. Manually change the destination file, after which Ascp and Aspera Sync can write content and permissions changes.
- When the destination file already exists and is immutable, Ascp and Aspera Sync return an error for the immutable file (`Destination: Read-only file system`) and then transfers any other files in the transfer.

Append-only files:

- When the source file is an append-only file (`appendOnly: yes`), it can be transferred once to the destination. After the initial transfer, it cannot be updated by Ascp or Aspera Sync because FASP must overwrite the file.
- When the destination file already exists and is an append-only file, Ascp and Aspera Sync either stop the rest of the transfer (for Spectrum Scale 5.0.1) or return an error for the file (`Destination: Read-only file system`) and then transfers any other files (for Spectrum Scale 5.0.0). To avoid transfer failures, Aspera recommends excluding append-only files that exist on the destination from consideration on the source.

Product Limitations

Describes any limitations that currently exist for Aspera transfer server and client products.

- **Path Limit:** The maximum number of characters that can be included in *any* pathname is 512 on Windows and 4096 on Unix-based platforms.
- **Illegal Characters:** Avoid the following characters in filenames: `/ \ " : ' ? > < & * |`.
- **Environment Variables:** The total size for environment variables depends on your operating system and transfer session. Aspera recommends that each environment variable value should not exceed 4096 characters.

Technical Support

Support Websites

For an overview of IBM Aspera Support services, go to https://asperasoft.com/company/support/.

To view product announcements, webinars, and knowledgebase articles, as well as access the Aspera Support Community Forum, sign into the IBM Aspera Support site at https://www.ibm.com/mysupport/ using your IBMid (not your company Aspera credentials), or set up a new account. Search for Aspera and select the product. Click **Follow** to receive notifications when new knowledgebase articles are available.

Personalized Support

You may contact an Aspera support technician 24 hours a day, 7 days a week, through the following methods, with a guaranteed 4-hour response time.

Email	aspera-support@ibm.com
Phone (North America)	+1 (510) 849-2386, option 2
Phone (Europe)	+44 (0) 207-993-6653 option 2
Phone (Singapore)	+81 (0) 3-4578-9357 option 2

Legal Notice