IBM Aspera HSTS 4.1

Contents

High-Speed Transfer Server Admin Guide for zLinux

Welcome to the High-Speed Transfer Server documentation, where you can find information about how to install, maintain, and use the High-Speed Transfer Server.

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.

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 HSTS, IBM Aspera High-Speed Transfer Endpoint (which permits one client connection),
- HSTS installed as part of IBM Aspera Faspex, or
- HSTS 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,
- HSTS and HSTE

What is FASP?

At the heart of your Aspera ecosystem are the FASP transfer engines Ascp and Ascp4. 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.

Ascp4 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 Ascp4 are installed and enabled with your installation of HSTS, HSTE, 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.

HSTS Web Portal

HSTS 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, HSTS 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 transfers.

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 HSTS or HSTE.

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 HSTS and HSTE, but both the client and server require a Aspera Sync-enabled license.

Get Started with an Aspera Transfer Server

As a server, HSTS is a remote endpoint that accepts authenticated connections from Aspera client applications and that participates as a source or destination for authorized transfers. Your server can also take the role of a client and connect to other Aspera servers to initiate transfers. The following steps describe how to prepare your system as a server.

Procedure

1. Review the system requirements and install HSTS.

   See “Requirements” on page 7 and “Installing HSTS” on page 9
2. Secure your server.

   For a compilation of Aspera-recommended security best practices, see “Aspera Ecosystem Security Best Practices” on page 346.

   a) Configure your firewall (see “Configuring the Firewall” on page 13).

   b) Change and secure the TCP port (see “Securing Your SSH Server” on page 14).

   c) Determine if you want to use server-side encryption at rest. See “Server-Side Encryption-at-Rest (EAR)” on page 103 for instructions on configuring this from the command line.

   You can also restrict user access to your server, which is described in a later step.
3. Add users and configure their access.

   Aspera client applications authenticate to the server using operating system accounts on the server. For example, if a remote client user, "marketing_mgr" wants to transfer with the server, add marketing_mgr as a system user on the server and then add marketing_mgr as an Aspera transfer user. To secure your server, restrict marketing_mgr's access to only certain directories on the server (set a docroot), set transfer permissions, and set the default shell as aspshell.

   a) For instructions on adding users, assigning users to aspshell, and setting a docroot, see “Setting Up Transfer Users ” on page 59.

   b) If you prefer to have your users authenticate to the server using SSH keys rather than with passwords, gather their public keys and install them on the server. For instructions, see “Setting Up a User's Public Key on the Server” on page 24.
4. Configure transfer settings and control bandwidth usage.

   Aspera FASP transfers can be configured globally, by group, or by user. You can set bandwidth caps and limit the total number of transfers. For more information on user-specific settings, see “aspera.conf - Transfer Server Configuration” on page 99.

   You can also set "virtual" bandwidth caps that can be assigned to incoming or outgoing transfers by group or by user. For more information, see “Controlling Bandwidth Usage with Virtual Links (Command Line)” on page 87.
5. Set up file validation and processing, if needed.

   You can protect your server against malicious software in uploaded files by using out-of-line file validation or inline file validation. For more information, see “Out-of-Transfer File Validation” on page 109 and “Inline File Validation” on page 111.

   You can configure your server to run other customized scripts at specific transfer events. For more information, see “Automated Execution of Lua Scripts with Transfer Events” on page 177.
6. Test that a remote client can access and transfer with your server.

For instructions, see “Testing a User-Initiated Remote Transfer” on page 25. If you have problems, review the topics in “Troubleshooting” on page 341.

Results

Once you confirm that remote clients can access your server, your basic server set up is complete.

- If you want to automatically distribute files and folders to clients when they are added to a specific folder on the server, see “Introduction to Watch Folders and the Aspera Watch Service” on page 183.
- If you want to enable server-based clients to synchronize files with your server, with the ability to synchronize bidirectionally, see “Introduction” on page 240.

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.

Procedure

1. Test a locally-initiated transfer to a server to confirm your installation and firewall configuration are operational.

   For instructions, see “Testing a Locally Initiated Transfer” on page 18. This provides a simple walk through of how to set up a connection with a server and transfer.
2. 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 152.
3. To run transfers from the command line, review the instructions for the Aspera command line clients.

   Your Aspera product comes with two command line clients: **ascp** and **ascp4**. They are similar but have different capabilities. For a comparison, see “Comparison of Ascp and Ascp4 Options” on page 157.

   - For more information about **ascp**, see “Ascp Command Reference” on page 121 and “Ascp General Examples” on page 136.
   - For more information about **ascp4**, see “Ascp4 Command Reference” on page 162 and “Ascp4 Examples” on page 170.

Results

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

- If you want to automatically distribute files and folders to clients when they are added to a specific folder on the server, see “Introduction to Watch Folders and the Aspera Watch Service” on page 183.
- If you want to synchronize files with your server, with the ability to synchronize bidirectionally, see “Introduction” on page 240. The **async** tool requires an additional license on each to run.

For a comparison of automatic transfer tools, see “Comparison of Aspera File Delivery and Synchronization Tools” on page 4.

Comparison of Aspera File Delivery and Synchronization Tools

Your Aspera product includes several transfer tools that can be used for automatic file delivery and synchronization.

- **Hot Folders:** a Windows-only, GUI-managed automatic file delivery tool.
- **Watch Folders:** an automatic file delivery tool that is easily managed by using the GUI, Console, or the Node API.

- **Aspera Sync:** a multi-directional synchronization tool for when complete file system synchronization is required.

	Hot Folders	**Watch Folders**	**Aspera Sync**
Supported platforms	Windows only	Windows macOS Linux AIX Solaris Linux on z Systems BSD	Windows macOS Linux AIX Solaris Linux on z Systems BSD
Additional license required	No	No	Yes, a Aspera Sync-enabled license is required on both endpoints
Interface	Aspera desktop GUI	Aspera desktop GUI, Node API in any command line, command line on the Aspera client, or Console web UI.	Aspera client command line, Console web UI for management only (no creation)
Client applications	HSTS and HSTE	HSTS and HSTE	HSTS and HSTE Drive
Server configuration required	No	No (only need asperawatchd on server for pull Watch Folders)	Recommended
Create in Console	No, but you can monitor transfers	Yes, you can create, monitor, and manage	No, but you can monitor Aspera Sync jobs and their associated transfer sessions
Transfer modes	• Client to server (push) • Server to client (pull)	• Client to server (push) • Server to client (pull)	• Client to server (push) • Server to client (pull) • Client and server (bidirectional)
File delivery or synchronization	File delivery: Files and folders added to or modified within a Hot Folder on the source are automatically sent to the destination folder. Files deleted from the source are not deleted on the destination.	File delivery: Files and folders added to or modified within a watch folder on the source are automatically sent to the destination folder. Files deleted from the source are not deleted on the destination.	Synchronization: All file system changes (additions, deletions, and modifications) are synchronized from source to destination (push or pull) or synchronized between source and destination (bidirectional).
File system monitoring	Windows operating system notifications.	File system snapshots collected by asperawatchd.	• In continuous mode: file system notifications • In scan (on-demand) mode: Aspera Sync scans the file system on the source side and

	Hot Folders	Watch Folders	Aspera Sync
			compares it to the Aspera Sync database • asperawatchd
Transfer schedules	• Immediate (as soon as a file system change in the Hot Folder is detected) • On a user-specified schedule	• Immediate (as soon as a difference between snapshots is detected)	• Immediate (in continuous mode or when using Aspera Sync with asperawatchd) • On a user-specified schedule (Aspera Sync run as a cron job)
Growing file support	No	Yes (on HSTS)	No
Database space requirements	None	At least 2 GB free per 1 million files, 3 GB free per 1 million files on Windows	At least 2 GB free per 1 million files, 3 GB free per 1 million files on Windows
Best for	• Automatic push and pull delivery with a simple GUI interface that does not require Console	• Automatic push and pull delivery with a simple GUI interface that does not require Console • Managing and monitoring push delivery through Console	• Precise synchronization between two endpoints of all file system changes (including deletions) • Bidirectional synchronization • Very large file sets - up to 100 million items across thousands of directories
Limitations	• Windows only • GUI must remain open • In pull mode, pull files even if they are in use	• Transfer rate of millions of small files can become limited by the speed at which database metadata can be written	• Continuous mode available only for Windows and Linux sources • Transfer rate of millions of small files can become limited by the speed at which database metadata can be written
More information	IBM Aspera High-Speed Transfer Server Admin Guide for Windows	“Introduction to Watch Folders and the Aspera Watch Service” on page 183	“Introduction” on page 240

Licenses and Entitlements

You must have either a perpetual license or consumption-based entitlement (also known as a metered license).

You will need your license or entitlement information when you do an installation or upgrade.

Note: Not all OS platforms support entitlements.

Perpetual Licenses

If you have purchased a perpetual license, you can obtain your license key, by emailing the IBM Aspera License Team at aspera-license@ibm.com with the site ID and the order number for your purchase, the part numbers and quantities needed, and the email addresses to which the licenses should be sent.

For information about installing your license, see “Installing HSTS” on page 9.

Consumption-Based Entitlements

If you have purchased a consumption-based entitlement (for Aspera on Demand or for tethered-node use with Aspera on Cloud), you will receive an email that includes your customer ID and entitlement ID. The email address of the IBM Aspera SaaS and On Demand Subscription Team is aspera-bss@wwpdl.vnet.ibm.com.

For information about setting up an entitlement, see “Installing HSTS” on page 9.

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 HSTS.

- Perpetual license.
- Supported operating system release (see release notes).
- SSH Server. Version 5.2 or higher is recommended.

To use the Node API:

- The line `127.0.0.1 localhost` must appear in the `hosts` file:

  `/etc/hosts`
- For UNIX-based nodes, SELinux must be set to `permissive` or `disabled`, not `enforced`. Check the status of SELinux with the following command:

  ```
  # sestatus
  ```

  If SELinux is set to `enforced`, see “Disabling SELinux” on page 344. If SELinux is set to `disabled` or `enforced`, or if **`sestatus`** reports that SELinux is not installed on your system, you do not need to take further action.
- To run node-to-node transfers, the remote node must have version 3.7.4 or later. Earlier versions use an SSH key type that is no longer accepted by servers as of version 3.7.4.

Before Upgrading or Downgrading

When upgrading or downgrading, Aspera recommends the following preparation to ensure a smooth process, minimal transfer disruption, and peace-of-mind that your original configuration is backed up in case of any problems.

About this task

Upgrading

- The installer automatically checks for an older version of the product on your system. If an older version is found, the installer automatically removes it before installing the new version.
- Although the installer performs your upgrade automatically, Aspera highly recommends completing the tasks below before upgrading. If you do not follow these steps, you risk installation errors or losing your former configuration settings.
- You cannot upgrade directly between different Aspera transfer products (such as from HSTS or HSTE). To upgrade, you need to back up the configuration, uninstall the product, and perform a fresh install of the new version of the product.

Downgrading

Older installers do not check for newer versions of the application. You must prepare your machine as described below then uninstall the newer version before continuing with your downgrade.

Note: Installers of versions 3.7.4 and older do not support `systemd`. When downgrading to one of these versions on an OS that uses `systemd`, you must manually start Aspera services because the installer cannot start them automatically. Run the following commands to start the services:

```
# systemctl start asperacentral
# systemctl start asperahttpd
# systemctl start asperarund
# systemctl start asperanoded
```

Newer versions of the Redis database are not compatible with older versions of the application. Your downgrade process depends on whether a backup of the older Redis DB is available, either as a separate backup file or as part of your backup of the `var` directory from the older version.

- **With a backup:** Follow the steps below to prepare your machine. Uninstall the application (for instructions, see "Uninstalling" on page 19). Copy the older Redis DB file into the `var` directory before installing the older (downgrade) version.

  `/opt/aspera/var/`
- **Without a backup:** Follow the steps below to prepare your machine. Uninstall the application (for instructions, see "Uninstalling" on page 19) and delete the `var` and `etc` directories from your machine. Then do a fresh installation of the older version. The configuration files in the `etc` directory may be compatible with older versions, but not all configurations may be read.

  `/opt/aspera/var/`

  `/opt/aspera/etc/`

Preparing for an Upgrade or Downgrade

Procedure

1. Verify the current version of HSTS.

   The steps that are required to prepare for an upgrade depend on your version. To view the current product and version, run the following command:

   ```
   # ascp -A
   ```
2. Review product release notes.

   Review the release notes for the versions that were released since your current version. In particular, the **Breaking Changes** section highlights changes that may require you to adjust your workflow, configuration, or usage.
3. If you were using asperawatchdor Watch Folders, set a docroot or restriction for the user running those services, if it is not already set.

   For more information on setting docroots or restrictions for users, see "Updating the Docroot or Restriction of a Running Watch Folder Service" on page 229. Ensure that the pathname being watched (the `source_dir` of the Watch Folder) is in the user's docroot or restriction.
4. If you were using asperawatchd or Watch Folders, prepare your Watch Folders for upgrade.

   Due to changes in the way watches are managed as of 3.8.0, the entire watch hierarchy is re-transferred after upgrade unless one of the following actions is taken to prepare your system:

   a. Archive files in the source directory before upgrade. This prevents asperawatchfolderd from considering all files in the source as new files and re-transferring them.

   b. Update the configuration of existing Watch Folders to set `"overwrite"` to NEVER. For instructions, see "Managing Watch Folders with aswatchfolderadmin" on page 213 or "Managing Watch Folders with the API" on page 223. After upgrade, Watch Folders only transfers files that do not exist at the target. Once the first drops complete, you can reset `"overwrite"` to your preferred setting.

5. Stop or allow to complete any FASP transfers that were initiated by the computer that you are upgrading.

   FASP transfers cannot proceed during your Aspera product upgrade.

   - Stop (and resume after upgrade) or allow to complete any Ascp, Ascp4, or Aspera Sync transfers in the command line.
6. Back up the Redis database.

   Stop asperanoded and create the Redis backup file by running the following commands:

   ```
   # systemctl stop asperanoded
   # /opt/aspera/bin/asredis -p 31415 BGREWRITEAOF
   ```

   The backup is stored as `appendonly.aof` in the following location:

   ```
   /opt/aspera/var/appendonly.aof
   ```
7. Back up configuration and settings files.

   These files are found in the `etc` and `var` folders.

   - `/opt/aspera/etc/`
   - `/opt/aspera/var/`
8. Back up the Redis database.

   The Redis database is backed up as part of backing up the `var` directory, but Aspera recommends backing it up separately as well, particularly if it is stored on a different machine.

   ```
   # sudo /opt/aspera/bin/asnodeadmin -b /filepath/database.backup
   ```
9. If you modified the daemon startup scripts for Aspera Central and asperanoded (for example, as part of an Aspera API integration), back up the modified files. These files are overwritten during an upgrade and you will need to copy your modifications into the new files after upgrading.

Installing HSTS

To install HSTS, log into your computer with root permissions.

About this task

Important: If this is a product upgrade, review all prerequisites described in “Before Upgrading or Downgrading” on page 7.

Procedure

1. Download HSTS from https://www.ibm.com/products/aspera/downloads.

   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 “Before Upgrading or Downgrading” on page 7 . 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-hsts`

OS	Commands
	-*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-hsts -version.rpm`

4. Installation troubleshooting.

   If the installer freezes during installation, another Aspera product might be running on your computer. To stop all FASP transfer-related applications and connections, see "Before Upgrading or Downgrading" on page 7.

5. Install the license.

   a) Create the Aspera license file and paste your license key string into it.

   ```
   /opt/aspera/etc/aspera-license
   ```

   b) Save and close the file.

   c) Verify that the license successfully installed:

   ```
   #
   ascp -A
   ```

   To update your product license after the installation, see "Updating a Perpetual License " on page 19.

6. If you plan to use Watch Folders, enable the services that allow asperarund (the service that manages Watch Folders) to automatically start after a reboot.

   For Debian OS, run the following commands:

   ```
   #
   systemctl enable systemd-networkd
   #
   systemctl enable systemd-networkd-wait-online.service
   ```

   For RedHat, zLinux, and CentOS, run the following commands:

   ```
   #
   systemctl enable NetworkManager
   #
   systemctl enable NetworkManager-wait-online.service
   ```

7. Edit OpenSSH authentication methods.

   a) Open your SSH Server configuration file from `/etc/ssh/sshd_config` with a text editor.

   b) To allow public key authentication, set `PubkeyAuthentication` to `yes`. To allow password authentication, set `PasswordAuthentication` to `yes`.

   For example,

   ```
   ... PubkeyAuthentication yes PasswordAuthentication yes ...
   ```

   c) Save the file then reload the SSH service.

d) Restart the SSH server to apply new settings.

Restarting your SSH server does not affect currently connected users.

```
# systemctl restart sshd.service
```

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

```
# service sshd restart
```

e) To further secure your SSH Server, see “Securing Your SSH Server” on page 14.

8. Secure your server or update your existing configuration.

For a compilation of Aspera-recommended security best practices, see “Aspera Ecosystem Security Best Practices” on page 346.

a) Configure your firewall (see “Configuring the Firewall” on page 13).

b) Change and secure the TCP port (see “Securing Your SSH Server” on page 14).

c) Determine if you want to use server-side encryption at rest. See “Server-Side Encryption-at-Rest (EAR)” on page 103 for instructions on configuring this from the command line.

Upgrade Follow up

Procedure

1. If you were using asperawatchd or Watch Folders in version 3.6.1 or earlier, manually migrate any services that are run by a user other than root.

The installer does not automatically migrate asperawatchd or asperawatchfolderd for users other than root, and you must manually start their services after upgrade:

a) Confirm that the user has a docroot set in `aspera.conf`.

To view the user's settings, run:

```
#
/opt/aspera/bin/
asuserdata -u user
```

If a value is not set for `absolute` in the `docroot option set` section, set a docroot by running the following command:

```
#
/opt/aspera/bin/
asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
```

b) Confirm that the user has permissions to write to the log directory.

To view the log directory settings, run:

```
#
/opt/aspera/bin/
asuserdata -a
```

Look for the values for `rund_log_dir` and `watch_log_dir`. If they are set to `"AS_NULL"`, then the logs write to the default directory (`/var/log/aspera.log`).

c) Start asperawatchd and asperawatchfolderd for the user by running the following commands:

```
#
/opt/aspera/sbin/
```

```
asperawatchd --user username

#
/opt/aspera/sbin/
asperawatchfolderd --user username
```

2. If you are updating an AoC node, restore the AoC data to the Redis database.
   a) Stop asperanoded.

```
#
systemctl stop asperanoded
```

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

```
#
service asperanoded stop
```

   b) Flush existing data from the Redis database on the new node.

```
#
/opt/aspera/bin/asredis -p 31415 FLUSHALL
```

   c) Load the backup database file into the new node database.

```
#
cat

/opt/aspera/bin/appendonly.aof | asredis --pipe -p 31415
```

   d) Restart asperanoded.

```
#
systemctl start asperanoded
```

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

```
#
service asperanoded start
```

3. If the Redis database is run on another system: Update the KV store keys to the latest format.
   The local Redis database schema is automatically updated by the installer, but non-local Redis databases must be manually updated by running the following command as root :

```
# /opt/aspera/bin/asnodeadmin --db-update
```

4. If you have a backup of modified daemon start up scripts for asperacentral and asperanoded, copy your modifications into the new versions of these scripts. Restart the services to activate your changes.
5. **For all upgrades:** Validate `aspera.conf`.
   The `aspera.conf` file is not overwritten during an upgrade and your configurations are preserved. However, the XML formatting, parameters, and acceptable values may have changed between your old

version and new version. Run the following command to check `aspera.conf` for XML form and valid configuration settings:

```
#
  /opt/aspera/bin/asuserdata -v
```

Configuring the Firewall

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

HSTS

Configure your firewall to allow the following ports:

- **Inbound TCP/22 (or other TCP port set for SSH connections):** The port for SSH connections.

  Important: Aspera strongly recommends running the SSH server on a non-default port (allowing inbound SSH connections on TCP/33001, and disallowing inbound connections on TCP/22) to ensure that your server remains secure from SSH port scan attacks. For instructions on how to change your SSH port, see “Securing Your SSH Server” on page 14.

  If you have a legacy customer base that uses TCP/22 then you can allow inbound connections on both ports. See “Securing Your SSH Server” on page 14 for instructions.

  The firewall on the server side must allow the open TCP port to reach HSTS. No servers are listening on UDP ports. When a transfer is initiated by an Aspera client, the client opens an SSH session to the SSH server on the designated TCP port and negotiates the UDP port for the data transfer.
- **Inbound UDP/33001:** The port for FASP transfers, which use UDP/33001 by default, although the server may also choose to run FASP transfers on another port.
- **Inbound and outbound TCP/8080 and TCP 8443 (or other TCP ports set for HTTP/HTTPS fallback):** The ports for the HTTP fallback. If only HTTP or HTTPS is used, you need to open only that port. For more information on configuring HTTP fallback ports, see “Ascp Command Reference” on page 121.
- **Local firewall:** If you have a local firewall on your server (like `iptables`), verify that it is not blocking your SSH and FASP transfer ports (such as TCP/UDP 33001). If you are using Vlinks, you will need to allow the Vlink UDP port (55001, by default) for multicast traffic. For additional information on setting up Vlinks, see “Controlling Bandwidth Usage with Virtual Links (Command Line)” on page 87.

Remote Client Machines

Typically, consumer and business firewalls allow direct outbound connections from client computers on TCP and UDP, and no configuration is required for Aspera transfers. In the special case of firewalls blocking direct outbound connections, usually with proxy servers for web browsing, the following ports must be allowed:

- **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 (or a range, if required):** 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).

Important: Multiple concurrent clients cannot connect to a Windows Aspera server on the same UDP port. Similarly, multiple concurrent clients that are utilizing two or more user accounts cannot connect to a macOS or FreeBSD Aspera server on the same UDP port. If connecting to these servers, you will need to allow a range of outbound connections from the Aspera client (that have been opened incrementally on

the server side, starting at UDP/33001). For example, you may need to allow outbound connections on UDP/33001 through UDP/33010 if 10 concurrent connections are allowed by the server.

Securing Your SSH Server

Keeping your data secure is critically important. Aspera strongly recommends taking additional steps to set up and configure your SSH server to protect against common attacks.

About this task

These steps include the following:

- Changing the TCP port.
- Configuring transfer server authentication.

Aspera also recommends restricting user access to the server, as described in the user setup instructions later in this guide.

Changing and Securing the TCP Port

SSH servers, including the OpenSSH suite included with your product, listen for incoming connections on TCP Port 22 by default. As such, Port 22 is subject to numerous unauthorized login attempts by hackers who attempt to access unsecured servers. An effective deterrent is to close Port 22 and run the service on a seemingly random port above 1024 (and up to 65535).

About this task

To standardize the port for use in Aspera transfers, Aspera recommends setting the TCP port to 33001 and closing TCP/22.

Prerequisites:

- Before changing the default port for SSH connections, verify with your network administrators that TCP/33001 is open.
- Before closing port TCP/22, notify users of the change.

Notifying Users - How to Specify TCP/33001

Aspera recognizes that disabling the default SSH connection port (TCP/22) might affect your clients. When you change the port, ensure that you advise your users on how to configure the new port number, from the GUI (if available and used) and from the command line.

- **GUI:** To change the SSH port in Desktop Client, click **Connections** and select the entry for the server whose ports are changing. On the **Connection** tab, click **Show Advanced Settings** and enter the SSH port number in the **SSH Port (TCP)** field.
- **Command line:** Clients running FASP transfers from the command line can specify the port by using the `-P 33001` option.

Changing to TCP/33001

The following steps require root privileges.

Procedure

1. Open the SSH configuration file.

   ```
   /etc/ssh/sshd_config
   ```

2. Add the TCP/33001 SSH port and close TCP/22.

   Comment out the line for "Port 22" and add a line for "Port 33001":

   ```
   #Port 22
   Port 33001
   ```

Once this setting takes effect:

- Aspera clients must set the transfer port to 33001 in the GUI or specify **`-P 33001`** for command line transfers.
- Server administrators should use `ssh -p 33001` to access the server through SSH.

3. Disable non-admin SSH tunneling.

   These instructions require that OpenSSH 4.4 or newer is installed on your system in order to use the `Match` directive. `Match` allows you to selectively override certain configuration options when specific criteria (based on user, group, hostname, or address) are met.

   Open your SSH Server configuration file, `sshd_config`, with a text editor. Add the following lines to the end of the file (or modify them if they already exist):

   ```
   AllowTcpForwarding no
   Match Group root
   AllowTcpForwarding yes
   ```

   Depending on your `sshd_config` file, you might have additional instances of `AllowTCPForwarding` that are set to the default `Yes`. Review your `sshd_config` file for other instances and disable if necessary.

   Disabling TCP forwarding does not improve security unless users are also denied shell access, because they can still install their own forwarders. Review your user and file permissions, and see “Setting Up Transfer Users ” on page 59 for instructions on modifying user shell access.

4. Update authentication methods

   Public key authentication can prevent brute-force SSH attacks if all password-based authentication methods are disabled. For this reason, Aspera recommends disabling password authentication in the `sshd_config` file and enabling private/public key authentication.

   Note: Before proceeding, configure at least one non-root, non-transfer user with a public key to use to manage the server. This is because in other server-securing steps, root login is disabled and Aspera recommends that transfer users are restricted to aspshell, which does not allow interactive login. This user and public key is what you use to access and manage the server as an administrator.

   To configure authentication methods, add or uncomment `PubkeyAuthentication yes` and comment out `PasswordAuthentication yes`.

   ```
   PubkeyAuthentication yes
   #PasswordAuthentication yes
   PasswordAuthentication no
   ```

   Note: If you choose to leave password authentication enabled, be sure to advise account creators to use strong passwords and set `PermitEmptyPasswords` to "no".

   ```
   PermitEmptyPasswords no
   ```

5. Disable root login.

   CAUTION: This step disables root access. Make sure that you have at least one user account with sudo privileges before continuing, otherwise you may not have access to administer your server.

   By default, OpenSSH allows root logins. However, disabling root access helps maintain a more secure server. Aspera recommends disabling root access by commenting out `PermitRootLogin yes` in the `sshd_config` file and adding `PermitRootLogin No`.

   ```
   #PermitRootLogin yes
   PermitRootLogin no
   ```

   Administrators can use the **su** command when root privileges are necessary.

6. Restart the SSH server to apply new settings.

Restarting your SSH server does not affect currently connected users.

```
# systemctl restart sshd.service
```

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

```
# service sshd restart
```

7. Review your logs periodically for attacks.

   You can view the state of active TCP connections by running the **netstat** command:

   ```
   # netstat -an -p tcp
   ```

   Typical output shows multiple, different IP addresses connected to specific ports:

   ```
   TCP    10.0.111.200:53402     72.21.81.109:80        CLOSE_WAIT
   TCP    10.0.111.200:53865     173.194.202.188:5228   ESTABLISHED
   TCP    10.0.111.200:53876     10.0.9.16:445          TIME_WAIT
   TCP    10.0.111.200:55164     208.85.40.20:443       ESTABLISHED
   TCP    10.0.111.200:55335     207.200.35.240:443     ESTABLISHED
   TCP    10.0.111.200:55444     67.199.110.81:443      ESTABLISHED
   TCP    10.0.111.200:56278     104.24.11.90:443       ESTABLISHED
   ```

   If your server is under attack, you might see output similar to the following, in which the same IP address attempts to connect to contiguous ports (hundreds or thousands of times) and the connection is timing out (reporting a status of `TIME_WAIT`):

   ```
   TCP    10.0.111.200:53402     72.21.81.109:60974      TIME_WAIT
   TCP    10.0.111.200:53865     72.21.81.109:60975      TIME_WAIT
   TCP    10.0.111.200:53876     72.21.81.109:60976      TIME_WAIT
   TCP    10.0.111.200:55164     72.21.81.109:60977      TIME_WAIT
   TCP    10.0.111.200:55335     72.21.81.109:60978      TIME_WAIT
   TCP    10.0.111.200:55444     72.21.81.109:60979      TIME_WAIT
   TCP    10.0.111.200:56278     72.21.81.109:60980      TIME_WAIT
   ```

   If you see this, review your logs to determine the source and cause.

   Open your syslog, which is located in `/var/log/auth.log` or `/var/log/secure`, depending on your system configuration.

   Look for invalid users in the log, especially a series of login attempts with common user names from the same address, usually in alphabetical order. For example:

   ```
   ...
   Mar 10 18:48:02 sku sshd[1496]: Failed password for invalid user alex from 1.2.3.4 port 1585
   ssh2
   ...
   Mar 14 23:25:52 sku sshd[1496]: Failed password for invalid user alice from 1.2.3.4 port
   1585 ssh2
   ...
   ```

   If you identify attacks, take the following steps:

   - Double-check the SSH security settings in this topic.
   - Report attackers to your ISP's email address for abuse reports (often `abuse@`*`your_isp`*`.com`).

Configuring Transfer Server Authentication With the Host-Key Fingerprint

To prevent server impersonation and man-in-the-middle (MITM) attacks, Aspera clients can verify the server's authenticity before starting a transfer by comparing the trusted SSH host key fingerprint (obtained directly from the server admin or through an Aspera client web application) with the host key fingerprint returned when the connection is made. In order to do this, the host key fingerprint must be set in both the system's `sshd_config` file and the server's `aspera.conf` file.

About this task

Procedure

1. Set the host key fingerprint in the server's `sshd_config` file (with `HostKey`).

   Note: Server SSL certificate validation (HTTPS) is enforced if a fingerprint is specified in `sshd_config` and HTTP fallback is enabled. If the transfer "falls back" to HTTP and the server has a self-signed certificate, validation fails. The client requires a properly signed certificate.

   If you set the host key path, the fingerprint is automatically extracted from the key file and you do not extract it manually.

   Retrieving and setting the host key fingerprint:

   a) Retrieve the server's SHA-1 fingerprint.

   ```
   # cat /etc/ssh/ssh_host_ecdsa_key.pub | awk '{print $2}' | base64 -d | sha1sum
   ```

   CAUTION:

   The client products listed below will prefer ECDSA SSH host keys when initiating a transfer with HSTS. If you are going to use the server with any of the following products, you must modify your configuration to use RSA encryption. If you do not, you will get an error for mismatched fingerprints, and your transfers will fail.

   - IBM Aspera Connect 3.10 or earlier
   - IBM Aspera Drive 3.2 or earlier
   - IBM Aspera Command-Line Interface (CLI) 3.9 or earlier
   - IBM Aspera Cargo 3.2 or earlier

   To use RSA encryption, you must:

   i) Edit `/etc/ssh/sshd_config`, as follows:

   ```
   HostKey /etc/ssh/ssh_host_rsa_key
       #HostKey /etc/ssh/ssh_host_ecdsa_key
       #HostKey /etc/ssh/ssh_host_ed25519_key
   ```

   ii) Restart the SSH server to apply new settings. The commands for System D and System V (init.d) systems are:

   ```
   # systemctl restart sshd.service
   ```

   ```
   # service sshd restart
   ```

   b) Set the SSH host key fingerprint in `aspera.conf`.

   ```
   # asconfigurator -x "set_server_data;ssh_host_key_fingerprint,fingerprint"
   ```

   This command creates a line similar to the following example of the `<server>` section of `aspera.conf`:

   ```
   <ssh_host_key_fingerprint>7qdOwebGGeDeN7Wv+2dP3HmWfP3
   </ssh_host_key_fingerprint>
   ```

   Setting the host key path: To set the SSH host key path instead of the fingerprint, from which the fingerprint will be extracted automatically, run the following command:

   ```
   # asconfigurator -x "set_server_data;ssh_host_key_path,ssh_key_filepath"
   ```

   This command creates a line similar to the following in the `<server>` section of `aspera.conf`:

   ```
   <ssh_host_key_path>/etc/ssh/ssh_host_ecdsa_key.pub
   </ssh_host_key_path>
   ```

2. Restart the node service to activate your changes.

Run the following commands to restart asperanoded:

```
# systemctl restart asperanoded
```

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

```
# service asperanoded restart
```

Testing a Locally Initiated Transfer

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

About this task

Procedure

1. Download test files from the server.

   For example, after using the following command to download, you would press y to accept the server's key, and enter the password when prompted:

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

   This transfer command is based on the following settings:

Item	Value
demo server address	`demo.example.com`
Login account	`mu_user_name`
password	`my_password`
Test file	`/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.
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.

   ```
   # ascp -T /tmp/100MB my_password@demo.example.com:Upload/
   ```

Updating a Perpetual License

Update your perpetual license from the command line.

Procedure

1. Open the license file with write permission.

```
/opt/aspera/etc/aspera-license
```

2. Replace the existing license key string with the new one, and save the file.
3. To confirm that the new license information has been updated correctly, run **ascp -A** to display the current license information.

```
# ascp -A
```

4. If you are using the Node API, reload asperanoded.

```
# /opt/aspera/bin/asnodeadmin --reload
```

Aspera Entitlement Service (ALEE) Configuration for Use with an HTTP Web Proxy

If you have a consumption-based entitlement (as opposed to a perpetual license), and you want to use an HTTP Web proxy, you must configure the proxy, or the entitlement service (ALEE) itself, depending on the proxy's features.

For example, if you are using a Squid proxy, add the following options to the JVM_OPTIONS assignment in :

```
" -Dhttp.proxyHost=10.110.12.44 -Dhttp.proxyPort=3128 -Dhttps.proxyHost=10.110.12.44 -
Dhttps.proxyPort=3128 "
```

Uninstalling

HSTS can be uninstalled without removing existing configuration files.

Procedure

1. If you are uninstalling in order to upgrade your Aspera product, review the upgrade preparation steps in “Before Upgrading or Downgrading” on page 7.
2. Close or stop the following applications and services:
   - FASP transfers
   - SSH connections
3. Uninstall HSTS by running the following command:

```
$ rpm -e ibm-aspera-entsrv
```

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

What to do next

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

Server Set up Methods

Users, groups, and transfers can be configured in several ways, all of which modify the server configuration file `aspera.conf`.

- **Running `asconfigurator` commands**

  Run **`asconfigurator`** commands from Terminal to automatically insert parameter settings as well-formed XML into `aspera.conf`. Use of asconfigurator commands is described in “Set up Users and Groups ” on page 59 and “Configure the Server from the Command Line” on page 67.

- **Manually editing `aspera.conf`**

  Open `aspera.conf` in a text editor with write permission and add or edit the text in XML format. Find `aspera.conf` in the following location:

  `/opt/aspera/etc/aspera.conf`

  For templates of `aspera.conf` parameter settings, see “Set up Users and Groups ” on page 59 and “Configure the Server from the Command Line” on page 67.

Set up Users and Groups

Aspera clients connect to HSTS by authenticating as a system user who is configured in the application. The user can also belong to a group that is configured in the application. Users and groups can be set up in the HSTS GUI.

Setting Up Users

HSTS uses system accounts to authenticate connections from Aspera clients. The system users must be added and configured as Aspera transfer users before clients can browse the server file system or run FASP transfers to and from the server. When creating transfer users, you can also specify user-specific settings, such as transfer bandwidth, docroot, and file handling. User configuration is an important part of securing your server. For a complete description, see “Aspera Ecosystem Security Best Practices” on page 346.

About this task

Important Configuration Notes:

- Some Aspera features require a docroot in URI format or require a file restriction instead of a docroot. For more information, see “Docroot vs. File Restriction” on page 345.
- If users connect to the server by providing IBM Aspera Shares credentials or by providing Node API credentials that are associated with the transfer user, changes to a user's configuration, such as their docroot, are not applied to the user until asperanoded is restarted. For instructions, see “Restarting Aspera Services” on page 344.

To configure a system user account as an Aspera transfer user:

Procedure

1. Restrict user permissions with **`aspshell`**.

   By default, all system users can establish a FASP connection and are only restricted by file permissions. Restrict the user's file operations by assigning them to use **`aspshell`**, which permits only the following operations:

   - Running Aspera uploads and downloads to or from this computer.

- Establishing connections in the application.
- Browsing, listing, creating, renaming, or deleting contents.

These instructions explain one way to change a user account or active directory user account so that it uses the **aspshell**; there may be other ways to do so on your system.

Run the following command to change the user login shell to **aspshell**:

```
# sudo usermod -s /bin/aspshell username
```

Confirm that the user's shell updated by running the following command and looking for `/bin/aspshell` at the end of the output:

```
# grep username /etc/passwd
username:x:501:501:...:/home/username:/bin/aspshell
```

Note: If you use OpenSSH, sssd, and Active Directory for authentication: To make `aspshell` the default shell for all domain users, first set up a local account for server administration because this change affects all domain users. Then open `/etc/sssd/sssd.conf` and change `default_shell` from `/bin/bash` to `/bin/aspshell`.

2. Launch HSTS as .
3. Click **Configuration** to open the configuration settings window.
4. For server security, configure **Global** settings to restrict users' transfer and system permissions.
   a) Set a global docroot (**Absolute Path**) to an empty folder or a part of the file system specific to each user.

      If there is a pattern in the docroot of each user, for example, *username*, you can use a substitutional string. This way you assign independent docroot to each user without setting a docroot for each user individually

Substitutional String	Definition	Example
`$(name)`	system user's name	`/sandbox/$(name)`
`$(home)`	system user's home directory	`$(home)/Documents`

   b) On the **Docroot** tab, set **Read Allowed**, **Write Allowed**, and **Browse Allowed** to **false**.
   c) On the **Authorization** tab, deny incoming and outgoing transfers by default, then enable transfers for individual users as required (described in a later step).
   d) On the **Authorization** tab, set the token encryption key to a string of at least 20 random characters.
   e) If your workflow allows, on the **Authorization** tab set **Content Protection Required** to **true**.

      This setting enforces client-side encryption-at-rest. For more information, see “Client-Side Encryption-at-Rest (EAR)” on page 156.
   f) On the **Authorization** tab, set **Encryption Allowed** to **AES-128**.

      By setting an encryption cipher, uploads to the server must use the specified encryption cipher or stronger. Setting to **any** allows encrypted and unencrypted transfers.
5. Add a system user.
   a) In **Server Configuration**, go to **Users**.
   b) Click ✚ to add a new user.
   c) Enter the username.

      Usernames cannot contain the "@" symbol, except when using the *user@domain* format. For additional information, see Product Limitations.
6. Set the user's docroot and transfer permissions.
   a) Set a user-specific docroot, if the global docroot is not adequate.

In the user's **Docroot** tab (**Configuration > Users >***username* **> Docroot**), select the **Override** box for **Absolute Path** and enter or select an existing path as the user's docroot -- for example, . When finished, click **OK** or **Apply**.

b) Set read, write, and browse permissions.

On the **Docroot** tab, set **Read allowed** to **true** to enable the user to download from their docroot on the server, set **Write allowed** to **true** to enable the user to upload to the server and move files within their docroot, and set **Browse allowed** to **true** to enable the user to browse files within their docroot. For maximum security, allow users the minimum permissions required for their workflow.

c) Set transfer permissions.

On the **Authorization** tab, set **Incoming Transfers** to **allow** to allow the user to upload to the server within their docroot and set **Outgoing Transfers** to **allow** to allow the user to download from the server from their docroot.

7. If you provided an Aspera license during installation (rather than an entitlement), ensure that the transfer user has read permissions on the Aspera license file (`aspera-license`) so that they can run transfers.

   The license file is found in: `/opt/aspera/etc/`

8. Configure group and user settings.

   Settings are located in the **Docroot**, **Authorization**, **Bandwidth**, **Network**, **File Handling** and **Precedence** tabs. User settings take precedence over group settings, which take precedence over global settings; for more information, see “Configuration Precedence” on page 23.

Category	Description
“Docroot, File Permission, and Growing Files Configuration” on page 26	The document root settings.
“Authorization Configuration” on page 28	Connection permissions, token key, and encryption requirements.
“Bandwidth Configuration” on page 33	Incoming and outgoing transfer bandwidth and policy settings.
“Network Configuration” on page 41	Network IP, port, and socket buffer settings.
“File Handling Configuration” on page 42	File handling settings, such as file block size, overwrite rules, and exclude pattern.

Setting Up Groups

Transfer settings can be applied to your system's user groups. If users within a group do not have individual transfer settings, then the group's transfer settings are applied. HSTS doesn't create user groups on the operating system for you, so you must ensure that the groups exist before adding them to your Aspera product.

About this task

Procedure

1. Identify or create the user group(s) that you would like to add.

   For information on creating user groups, see your operating system documentation.

2. Launch the Aspera server as .
3. Click **Configuration** to open the configuration settings window.
4. Add the user group to your Aspera server.

   In the **Server Configuration** window, click the **Groups** tab then click + and input the group's name.

5. Configure the group's transfer settings.

These settings are located in the **Docroot**, **Authorization**, **Bandwidth**, **Network**, **File Handling** and **Precedence** tabs.

Settings	Description
“Docroot, File Permission, and Growing Files Configuration” on page 26	The document root settings.
“Authorization Configuration” on page 28	Connection permissions, token key, and encryption requirements.
“Bandwidth Configuration” on page 33	Incoming and outgoing transfer bandwidth and policy settings.
“Network Configuration” on page 41	Network IP, port, and socket buffer settings.
“File Handling Configuration” on page 42	File handling settings, such as file block size, overwrite rules, and exclude pattern.
“Configuration Precedence” on page 23	When a user is a member of multiple groups, the precedence setting can be used to determine priority.

Configuration Precedence

HSTS applies configuration settings in this order: 1) user settings, 2) group settings (and if a user belongs to more than one group, precedence can be set for each group), 3) global settings, 4) default settings. User settings have the highest priority and default the lowest.

For example, the table below shows the setting values that are applied to user `aspera_user_1` in **bold** when that user is also a member of several groups and global settings are configured. In this example, `aspera_user_1` is a member of both the **admin** and **xfer** groups. The **admin** group's precedence setting is 0, which supersedes the **xfer** group's setting of 1:

Options	"aspera_user_1" User Settings	"admin" Group Settings	"xfer" Group Settings	Global Settings	Default Settings
Target rate	**5M**	10M	15M	40M	45M
Min rate	n/a	**2M**	8M	3M	0
Policy	n/a	n/a	**Low**	Fair	Fair
Docroot	n/a	n/a	n/a	**/pod/$(name)**	n/a
Encryption	n/a	n/a	n/a	n/a	**any**

Configuring Precedence of Groups

You can configure a group's precedence in `aspera.conf` by running the following **`asconfigurator`** command:

```
# asconfigurator -x "set_group_data;group_name,group_name;precedence,value"
```

Note: A group's precedence setting must be greater than or equal to 0, where 0 is the highest precedence level.

This adds a <group> section to `aspera.conf` like the one below. In this example, group "admin" has higher precedence than group "xfer".

```
<groups>
   <group>
      <name>admin</name>
      <precedence>0</precedence>
      ...
```

```
    </group>
    <group>
       <name>xfer</name>
       <precedence>1</precedence>
       ...
    </group>
 </groups>
```

You can also edit `aspera.conf` manually by opening it with administrative privileges:

`/opt/aspera/etc/aspera.conf`

In the file, locate the entry for each group, add the <precedence> option, and assign a precedence value as shown in the example above. After editing the file, validate the XML form and option values by running the following command:

```
# /opt/aspera/bin/asuserdata -v
```

Setting Up a User's Public Key on the Server

Public key authentication is an alternative to password authentication, providing a more secure authentication method that allows users to avoid entering or storing a password, or sending it over the network. An Aspera client generates a key pair (a public key and a private key) on the client computer and provides the public key to the administrator of the remote Aspera transfer server. The server administrator sets up the client user's public key as described in the following steps.

About this task

For information on how to create public keys, see “Creating SSH Keys ” on page 152.

Procedure

1. Obtain the client user's public key.

   The client user should send you a secure email with the public key pasted in the message body or attached as a text file.
2. Install the public key in the user account on the server.
   a) In the home directory of the account that the client will use to access the server, create a directory called `.ssh` if it doesn't already exist.
   b) Save the key file as `authorized_keys` in `.ssh`. If `authorized_keys` already exists, append the key file to it.

      For example,

      ```
      # mkdir /home/aspera_user_1/.ssh
      # cat /tmp/id_rsa.pub > /home/aspera_user_1/.ssh/authorized_keys
      ```

      Where:

      - `aspera_user_1` is the server user account.
      - `/tmp/id_rsa.pub` is where you saved the public key sent by the user.
      - `/home/aspera_user_1/.ssh/authorized_keys` is the file that contains the public key.

   c) Configure permissions on the key.

      Make the system user (in this example, user `aspera_user_1`) the owner of key directory and key file, allow access by the `aspera_user_1` group, and set permissions:

      ```
      # chown -R aspera_user_1:aspera_user_1  /home/aspera_user_1/.ssh
      # chmod 700 /home/aspera_user_1
      # chmod 700 /home/aspera_user_1/.ssh
      # chmod 600 /home/aspera_user_1/.ssh/authorized_keys
      ```

Testing a User-Initiated Remote Transfer

Once you have configured an Aspera transfer user on HSTS, test that an Aspera client can successfully connect to HSTS and upload a file.

About this task

Prerequisites:

- **Client:** Install an Aspera client application, such as the freely available IBM Aspera Desktop Client or IBM Aspera Command-Line Interface, on the client computer.
- **Server:** HSTS must have at least one Aspera transfer user (a system user account that is configured to authenticate Aspera transfers) configured on it.

If any of the following connection tests fail, see "Clients Cannot Establish Connection" on page 341.

Procedure

1. On the client, test that you can reach the IP address of your server.

   Run the **ping** command:

   ```
   # ping 10.0.0.2
   PING 10.0.0.2 (10.0.0.2): 56 data bytes
   64 bytes from 10.0.0.2: icmp_seq=0 ttl=64 time=8.432 ms
   64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=7.121 ms
   64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=5.116 ms
   64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=4.421 ms
   64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.050 ms
   ...
   ```

   In the example above, the address of HSTS is 10.0.0.2 and the output shows successful responses from the host.

   If the output returns "Destination host unreachable," check the firewall configuration of the server.

2. On the client, try a transfer to HSTS by using **ascp**.

   Run the following command on your client machine:

   ```
   # ascp -P 33001 --mode=send --policy=fair -l 10000 -m 1000 source_path
   username@ip_address:destination
   ```

   For example: (where `aspera_user_1` is the example transfer user):

   ```
   # ascp -P 33001 --mode=send --policy=fair -l 10000 -m 1000 /client-dir/files
   aspera_user_1@10.0.0.2:/dir
   ```

   This command specifies the following values for the transfer:

Item	Argument	Example Value
TCP Port Set the TCP port to start the transfer session.	`-P port`	`-P 33001`
Transfer Direction Specify if the server is the destination or source.	`--mode=direction`	`--mode=send`
Transfer Policy Specify how to share bandwidth with other network users.	`--policy=policy`	`--policy=fair`

Item	Argument	Example Value
Maximum Transfer Rate Set the maximum transfer rate, in Kbps.	`-l` *rate*	`-l 10000` Maximum transfer rate = 10 Mbps
Minimum Transfer Rate Set the minimum transfer rate, in Kbps.	`-m` *rate*	`-l 1000` Minimum transfer rate = 1 Mbps
File or Directory to Upload Set the path relative to your current directory.	*source*	`/client-dir/files`
Transfer User	*username*	`aspera_user_1`
Host Address	*ip_address*	`10.0.0.2`
Destination Folder Set the destination path relative to the transfer user's docroot.	*destination*	`/dir` In this example, the files are transferred to the "dir" folder in the docroot of aspera_user_1.

Configure HSTS in the GUI

The following references describe the server settings that can be configured in the HSTS GUI. Not all settings are available in the GUI; some must be set by using the command line or directly editing the HSTS configuration file, `aspera.conf`.

Docroot, File Permission, and Growing Files Configuration

The **Docroot** configuration options include the docroot and file permissions. The absolute path, or docroot, is the area of the file system that is accessible to an Aspera transfer user. The default empty value allows access to the entire file system. You can set one global docroot and then further restrict access to the file system by group or individual user.

Important Configuration Notes:

- The default server configuration gives users full access to the server's file system with read, write, and browse privileges. Aspera strongly recommends setting a global docroot that is an empty folder and setting global file permissions to **false**. For a compilation of server security best practices, see “Aspera Ecosystem Security Best Practices” on page 346.
- Some Aspera features require a docroot in URI format or require a file restriction instead of a docroot. For more information, see “Docroot vs. File Restriction” on page 345.

1. Open HSTS with root privileges.
2. Click **Configuration > Docroot**.
3. Edit **Global**, **Groups**, and **Users** settings on their **Docroot** tabs. Select **Override** in the option's row to set an effective value. User settings take precedence over group settings, which take precedence over global settings.

   Aspera recommends setting restrictive **Global** settings, as described in the following table, and then granting permissions for specific **Groups** or **Users**.

Docroot Settings Reference

Field	Description	Values	Default
Absolute Path	The absolute path, or docroot, is the area of the file system that is accessible to an Aspera transfer user. The default empty value allows access to the entire file system. You can set one global docroot and then further restrict access to the file system by group or individual user. Docroot paths require specific formatting depending on where the transfer server's storage is located. **Format examples** • Local storage absolute path:`/home/aspera424/movies` Or using a placeholder for usernames: `/home/$(name)` • Local storage in URI format: `file:///home/bear/movies` URI format is required for server-side encryption-at-rest, but is not supported by the Aspera Watch Service. Aspera recommends setting a global docroot to an empty folder or a part of the file system specific to each user. If there is a pattern in the docroot of each user, for example, *`username`*, you can use a substitutional string. This allows you to assign an independent docroot to each user without setting it individually for each user. See “Setting Up Users” on page 20 for information. You can also set multiple docroots and make them conditional based on the IP address from which the connection is made by editing `aspera.conf`. To do so, edit the absolute path setting by adding the IP address using the following syntax: `<absolute peer_ip="ip_address">path</absolute>` Growing files support allows you to start transferring files to the target directory while they are still being written to the source directory. To configure `aspera.conf` for growing files: • Edit the `<absolute>` section. • Add your growing files specification using the syntax described in “Ascp Command Reference” on page 121 for the *source* element. See also “Ascp General Examples” on page 136.	file path or URI	undefined (total access)
Read Allowed	Set to `true` (default) to allow users to transfer files and folders from their docroot.	• `true` • `false`	`true`

Field	Description	Values	Default
Write Allowed	Set to `true` (default) to allow users to transfer files and folders to their docroot.	• `true` • `false`	`true`
Browse Allowed	Set to `true` (default) to allow users to browse their docroot.	• `true` • `false`	`true`

Authorization Configuration

The **Authorization** configuration options include connection permissions, token key, and encryption requirements.

Note: For security, Aspera recommends denying incoming and outgoing transfers globally, then allowing transfers by individual users, as needed. For a compilation of server security best practices, see "Aspera Ecosystem Security Best Practices" on page 346.

1. Open the application with root privileges.
2. Click **Configuration > Authorization**.
3. Edit **Global** , **Groups**, and **Users** settings on their **Authorization** tabs. Select **Override** in the option's row to set an effective value. User settings take precedence over group settings, which take precedence over global settings.

Authorization Settings Reference

Setting	Description	Values	Default
Incoming Transfers	To enable users to transfer to this computer, leave the default setting of `allow`. Set to `deny` to prevent transfers to this computer. Set to `token` to allow only transfers initiated with valid tokens to this computer. Token-based transfers are typically used by web applications such as IBM Aspera Faspex and IBM Aspera Shares and require a Token Encryption Key.	`allow`, `deny`, or `token`	`allow`
Incoming External Provider URL	Set the URL of the external authorization provider for incoming transfers. The default empty setting disables external authorization. Aspera servers can be configured to check with an external authorization provider. This SOAP authorization mechanism can be useful to organizations requiring custom authorization rules. Requires a value for Incoming External Provider SOAP Action.	HTTP URL	blank
Incoming External Provider SOAP Action	The SOAP action required by the external authorization provider for incoming transfers. Required if Incoming External Provider URL is set.	text string	blank
Outgoing Transfers	To enable users to transfer from this computer, leave the default setting of `allow`. Set to `deny` to prevent transfers from this computer. Set to `token` to allow	`allow`, `deny`, or `token`	`allow`

Setting	Description	Values	Default
	only transfers initiated with valid tokens from this computer. Token-based transfers are typically used by web applications such as Faspex and require a Token Encryption Key.		
Outgoing External Provider URL	Set the URL of the external authorization provider for outgoing transfers. The default empty setting disables external authorization. HSTS can be configured to check with an external authorization provider. This SOAP authorization mechanism can be useful to organizations requiring custom authorization rules. Requires a value for Outgoing External Provider Soap Action.	HTTP URL	blank
Outgoing External Provider Soap Action	The SOAP action required by the external authorization provider for outgoing transfers. Required if Outgoing External Provider URL is set.	text string	blank
Token Encryption Cipher	Set the cipher used to generate encrypted transfer tokens.	`aes-128`, `aes-192`, or `aes-256`	`aes-128`
Token Encryption Key	Set the secret text phrase that is used to authorize those transfers configured to require token. Aspera recommends setting a token encryption key of at least 20 random characters. For more information, see "Require Token Authorization: Set from the Command Line" on page 310.	text string	blank
Token Life (seconds)	Set the token expiration for users of web-based transfer applications.	positive integer	`86400` (24 hrs)
Strong Password Required for Content Encryption	Set to `true` to require that the password for content encryption (client-side encryption at rest) includes at least 6 characters, of which at least 1 is non-alphanumeric, at least 1 is a letter, and at least 1 is a digit.	`true` or `false`	`false`
Content Protection Secret	Enable server-side encryption-at-rest (EAR) by setting the passphrase. Files uploaded to the server are encrypted while stored there and are decrypted when they are downloaded. For more information, see "Server-Side Encryption-at-Rest (EAR)" on page 103.	passphrase	(none)
Content Protection Required	Set to `true` to require that uploaded content be encrypted by the client (enforce client-side encryption-at-rest). For more information, see "Client-Side Encryption-at-Rest (EAR)" on page 156.	`true` or `false`	`false`

Setting	Description	Values	Default
	Important: When a transfer falls back to HTTP or HTTPS, content protection is no longer supported. If HTTP fallback occurs while downloading, then–despite entering a passphrase–the file remains encrypted. If HTTP fallback occurs during upload, then–despite entering a passphrase–the files are not encrypted.		
Allow transfer when client lacks GCM	By default, when a server is configured for a GCM cipher (for example aes-256-gcm), and the client is running a server version 3.8 or lower, the transfer will fail because clients running version 3.8 or lower do not support GCM mode. However, setting `<strict_allowed_cipher>` to `false` will permit transfers under these conditions.	`true` or `false`	`true`
Do encrypted transfers in FIPS-140-2-certified encryption mode	Set to `true` for **ascp** to use a FIPS 140-2-certified encryption module. When enabled, transfer start is delayed while the FIPS module is verified. When you run **ascp** in FIPS mode (that is, `<fips_enabled>` is set to `true` in `aspera.conf`), and you use passphrase-protected SSH keys, you must use keys generated by running **ssh-keygen** in a FIPS-enabled system, or 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` **Important:** When set to `true`, all ciphers and hash algorithms that are not FIPS compliant will abort transfers.	`true` or `false`	`false`
Encryption Allowed	Set the transfer encryption allowed by this computer. Aspera strongly recommends that you require transfer encryption. 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. **Note:** To ensure client compatibility when requiring encryption, use a cipher with the form `aes-XXX`, which is supported by all clients and servers. Requiring GCM causes the server to reject transfers from clients	`any`, `none`, `aes-128`, `aes-192`, `aes-256`, `aes-128-cfb`, `aes-192-cfb`, `aes-256-cfb`, `aes-128-gcm`, `aes-192-gcm`, or `aes-256-gcm`	`any`

Setting	Description	Values	Default
	that are running a version of Ascp 3.8 or older, unless <strict_allowed_cipher> is set to false. When a client requests a shorter cipher key than is configured on the server (or in an access key that authorizes the transfer), the transfer is automatically upgraded to the server setting. For more information about how the server and client negotiate the transfer cipher, see the description of -c in "Ascp Command Reference" on page 121 and "Ascp4 Command Reference" on page 162. Values: • any - allow transfers that use any encryption cipher or none. • none - require unencrypted transfers (not recommended). • aes-128, aes-192, or aes-256 - allow transfers that use an encryption cipher key that is as long or longer than the setting. These settings use the CFB or GCM mode depending on the client version and cipher requested. Supports all client versions. • aes-128-cfb, aes-192-cfb, or aes-256-cfb - require that transfers use the CFB encryption mode and a cipher key that is as long or longer than the setting. Supports all client versions. • aes-128-gcm, aes-192-gcm, or aes-256-gcm - require that transfers use the GCM encryption mode introduced in version 3.9.0 and a cipher that is as long or longer than the setting.		
Allow transfer when client lacks GCM	By default, when a server is configured for a GCM-mode cipher (for example aes-256-gcm), and the client is running a version of Ascp 3.8 or older, the transfer will fail. However, setting <strict_allowed_cipher> to false will permit transfers under these conditions.	true or false	false

Server-Side Encryption at Rest (EAR)

Capabilities

Server-side EAR provides the following advantages:

- It protects files against attackers who might gain access to server-side storage. This is important primarily when using NAS storage or cloud storage, where the storage can be accessed directly (and not just through the computer running HSTS or HSTE).
- It is especially suited for cases where the server is used as a temporary location–for example, when a client uploads a file and another one downloads it.

- Server-side EAR can be used together with client-side EAR. When used together, content is doubly encrypted. For more information, see “Client-Side Encryption-at-Rest (EAR)” on page 156.
- Server-side EAR doesn't create an "envelope" as client-side EAR does. The transferred file stays the same size as the original file. The server stores the metadata necessary for server-side EAR separately in a file of the same name with the file extension `.aspera-meta`. By contrast, client-side EAR creates a envelope file containing both the encrypted contents of the file and the encryption metadata, and it also changes the name of the file by adding the file extension `.aspera-env`.
- It works with both regular transfers (FASP) and HTTP fallback transfers.

Limitations and Considerations

- Server-side EAR is not designed for cases where files need to move in an encrypted state between multiple computers. For that purpose, client-side EAR is more suitable: files are encrypted when they first leave the client, then stay encrypted as they move between other computers, and are decrypted when they reach the final destination and the passphrase is available.
- Server-side EAR does not work with multi-session transfers (using **`ascp -C`** or Node API `multi_session` set to greater than 1).
- Do not mix server-side EAR and non-EAR files in transfers, which can happen if server-side EAR is enabled after the server is in use or if multiple users have access to the same area of the file system but have different EAR configurations.

Configuring Server-Side EAR

Procedure

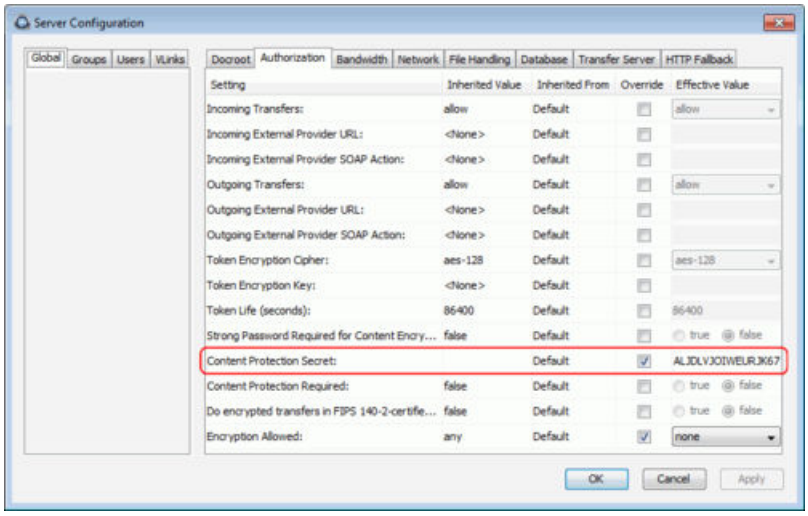
1. Set the docroot in URI format.

   Server-side EAR requires the storage to have a docroot in URI format, such that it is prefixed with `file:///`. The third slash (`/`) does not serve as the root slash for an absolute path. For example, a docroot of `/home/xfer` would be set as `file:////home/xfer` and a docroot of `C:\Users\xfer` would be set as `file:///C:\Users\xfer`.

   To configure the docroot options, click **Configuration** and set configurations for **Global**, **Groups**, or **Users** under their respective **Docroot** tabs. Select **Override** in the setting row to set a docroot and adjust read, write, and browse privileges. User docroot settings take precedence over group settings, which take precedence over global settings.

2. Set the password.

   The server-side EAR password can be set for all users (global), per group, or per user. In the **Server Configuration** dialog, click the **Authorization** tab and locate the setting for **Content Protection Secret**. Select the override box and enter the password.



Bandwidth Configuration

The **Bandwidth** configuration options include target transfer rates, transfer policies, and assigning vlinks to control aggregate bandwidth usage.

1. Open the application with root privileges.
2. Click **Configuration > Bandwidth**.
3. Edit **Global** , **Groups**, and **Users** settings on their **Bandwidth** tabs. Select **Override** in the option's row to set an effective value. User settings take precedence over group settings, which take precedence over global settings.

Bandwidth Settings Reference

Field	Description	Values	Default
Incoming Vlink ID	The ID of the vlink to apply to incoming transfers. Vlinks are a way to define aggregate transfer policies. For more information, see “Controlling Bandwidth Usage with Virtual Links (Command Line)” on page 87.	Vlink IDs	`Undefined` (Disabled)
Incoming Target Rate Cap (Kbps)	The maximum target rate for incoming transfers, in kilobits per second. No transfer session can exceed this rate at any time. If the client requests an initial rate greater than the target rate cap, the transfer proceeds at the target rate cap. The default setting of `unlimited` applies no target rate cap.	positive integer	unlimited
Incoming Target Rate Default (Kbps)	The default initial rate for incoming transfers, in kilobits per second. If allowed ("Incoming Target Rate Lock" is set to `false`), clients can modify this rate in real time. This setting is not relevant to transfers with a `fixed` bandwidth policy.	positive integer	`10000`

Field	Description	Values	Default
Incoming Target Rate Lock	Lock the target rate of incoming transfers to the default value (set to `true`). Set to `false` to allow users to adjust the transfer rate of an incoming transfer up to the "Incoming Target Rate Cap".	`true` or `false`	`false`
Incoming Minimum Rate Cap (Kbps)	The highest minimum rate that an incoming transfer can request, in kilobits per second. Client minimum rate requests that exceed the minimum rate cap are ignored. The default value of `unlimited` applies no cap to the minimum rate. **Important:** Aspera strongly recommends setting the minimum rate cap to zero. Transfers do not slow below the client's requested minimum rate unless the minimum rate is capped on the server. If the client-requested minimum rate exceeds network or storage capacity, this can decrease transfer performance and cause problems on the target storage.	positive integer or `unlimited`	`unlimited`
Incoming Minimum Rate Default (Kbps)	The default initial minimum rate for incoming transfers, in kilobits per second. If allowed ("Incoming Minimum Rate Lock" is set to `false`), clients can modify the minimum rate in real time, up to the "Incoming Minimum Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy.	positive integer	`0`
Incoming Minimum Rate Lock	Lock the minimum rate of incoming transfers to the default value (set to `true`). Set to `false` to allow users to adjust the minimum transfer rate up to the "Incoming Minimum Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy. **Important:** Aspera strongly recommends setting a lock on minimum rate to prevent transfers from using minimum rates that can overwhelm network or storage capacity, decrease transfer performance, and cause problems on the target storage.	`true` or `false`	`false`
Incoming Bandwidth Policy Allowed	The bandwidth policies that incoming transfers can use. Aspera transfers can use high, fair, low, or fixed bandwidth policies to determine bandwidth allocation among transfers. • `any` - The server does not deny any transfer based on policy setting.	`high`, `fair`, `low`, or `any`	`any`

Field	Description	Values	Default
	Note: Setting to any allows clients to request a `fixed` bandwidth policy. If the client also requests a high minimum transfer rate and that is not capped by the server, the transfer rate can exceed network or storage capacity. This can decrease transfer performance and cause problems on the target storage. To avoid these problems, set the allowed policy to `fair`. • `high` - Transfers that use `high`, `fair`, or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected. • `fair` - Transfers that use `fair` or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected. • `low` - Only transfers that use a `low` bandwidth policy are allowed. All others are rejected.		
Incoming Bandwidth Policy Default	The default bandwidth policy for incoming transfers. Clients can override the default policy if they specify a policy allowed by the server (see "Incoming Bandwidth Policy Allowed") and if "Incoming Bandwidth Policy Lock" is set to `false`. • `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.	`high`, `fair`, `low`, `fixed`	`fair`

Field	Description	Values	Default
	• `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.		
Incoming Bandwidth Policy Lock	Lock the bandwidth policy of incoming transfer sessions to the default value (set to `true`). Set to `false` to allow users to adjust the bandwidth policy.	`true` or `false`	`false`
Incoming Rate Control Module	Set how the transmission rate should be managed relative to instantaneous network bandwidth availability. Aspera recommends that this option be changed only by advanced users. When the client does not specify a configuration, the server configuration is used. When the client specifies a value other than `delay` and the client is the receiver, then the client configuration overrides the server configuration. Values: • **delay:** The baseline rate control module used by Aspera transfers. • **delay-odp:** A queue-scaling controller for overdrive protection. • **delay-adv:** An advanced rate controller. • **delay-laq:** A loss-adjusted queueing (LAQ) rate controller. **Note:** The LAQ module is an experimental rate control module that is designed to solve issues with target rate overdrive, high concurrency (when many FASP sessions run at the same time), and shallow buffers (limited packet queuing capability of a router). When LAQ is set, then it uses the FD31 RTT predictor unless a different RTT predictor is explicitly set. To set a rate control module for outgoing traffic, set it from the command line ("aspera.conf - Transfer Configuration" on page 70).	`delay`, `delay-odp`, `delay-adv`, or `delay-laq`	`delay`
Incoming Traffic RTT Predictor	The type of predictor to use to compensate for feedback delay when	`unset`, none, `alphabeta`,	`unset`

Field	Description	Values	Default
	measuring RTT. An experimental feature that might increase transfer rate stability and throughput by predicting network congestion. When set to `unset`, the client-specified predictor is used and if the client does not specify a predictor, then none is used. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90.	`fd31`, `bezier`, `ets`	
Incoming Rate Control Target Queue	The method for calculating the target queue. Static queuing is good for most internet connections, whereas dynamic queuing is good for satellite and other radio connections. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90. When set to `unset`, the client-specified transfer queuing method is used and if the client does not specify a queuing method, then `static` is used.	`unset`, `static`, `dynamic`	`unset`
Outgoing Vlink ID	The ID of the vlink to apply to outgoing transfers. Vlinks are a way to define aggregate transfer policies. For more information, see "Controlling Bandwidth Usage with Virtual Links (Command Line)" on page 87.	Vlink ID	`Undefined` (Disabled)
Outgoing Target Rate Cap (Kbps)	The maximum target rate for outgoing transfers, in kilobits per second. No transfer session can exceed this rate at any time. If the client requests an initial rate greater than the target rate cap, the transfer proceeds at the target rate cap. The default setting of `unlimited` applies no target rate cap.	positive integer	`unlimited`
Outgoing Target Rate Default (Kbps)	The default initial rate for outgoing transfers, in kilobits per second. If allowed ("Outgoing Target Rate Lock" is set to `false`), clients can modify this rate in real time up to the "Outgoing Target Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy.	positive integer	`10000`
Outgoing Target Rate Lock	Lock the target rate of outgoing transfers to the default value (set to `true`). Set to `false` to allow users to adjust the transfer rate of an outgoing transfer.	`true` or `false`	`false`
Outgoing Minimum Rate Cap (Kbps)	The highest minimum rate that an outgoing transfer can request, in kilobits per second. Client minimum rate requests that exceed the minimum rate cap are ignored. The default value of	positive integer	`unlimited`

Field	Description	Values	Default
	`unlimited` applies no cap to the minimum rate. **Important:** Aspera strongly recommends setting the minimum rate cap to zero. Transfers do not slow below the client's requested minimum rate unless the minimum rate is capped on the server. If the client-requested minimum rate exceeds network or storage capacity, this can decrease transfer performance and cause problems on the target storage.		
Outgoing Minimum Rate Default	The default initial minimum rate for outgoing transfers, in kilobits per second. If allowed ("Outgoing Minimum Rate Lock" is set to `false`), clients can modify the minimum rate in real time up to the "Outgoing Minimum Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy.	positive integer	`0`
Outgoing Minimum Rate Lock	Lock the minimum rate of outgoing transfers to the default value (set to `true`). Set to `false` to allow users to adjust the minimum transfer rate. This setting is not relevant to transfers with a `fixed` bandwidth policy. **Important:** Aspera strongly recommends setting a lock on minimum rate to prevent transfers from using minimum rates that can overwhelm network or storage capacity, decrease transfer performance, and cause problems on the target storage.	`true` or `false`	`false`
Outgoing Bandwidth Policy Allowed	The bandwidth policies that outgoing transfers can use. Aspera transfers can use high, fair, low, or fixed bandwidth policies to determine bandwidth allocation among transfers. • `any` - The server does not deny any transfer based on policy setting. **Note:** Setting to `any` allows clients to request a `fixed` bandwidth policy. If the client also requests a high minimum transfer rate and that is not capped by the server, the transfer rate can exceed network or storage capacity. This can decrease transfer performance and cause problems on the target storage. To avoid these problems, set the allowed policy to `fair`.	`high`, `fair`, `low`, or `any`	`any`

Field	Description	Values	Default
	• `high` - Transfers that use `high`, `fair`, or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected. • `fair` - Transfers that use `fair` or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected. • `low` - Only transfers that use a `low` bandwidth policy are allowed. All others are rejected.		
Outgoing Bandwidth Policy Default	The default bandwidth policy for outgoing transfers. Clients can override the default policy if they specify a policy allowed by the server (see "Outgoing Bandwidth Policy Allowed") and if "Outgoing Bandwidth Policy Lock" is set to `false`. • `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.	`high`, `fair`, `low`, `fixed`	`fair`
Outgoing Bandwidth Policy Lock	Lock the bandwidth policy of outgoing transfer sessions to the default value (set	`true` or `false`	`false`

Field	Description	Values	Default
	to `true`). Set to `false` to allow users to adjust the bandwidth policy.		
Outgoing Traffic RTT Predictor	The type of predictor to use to compensate for feedback delay when measuring RTT. An experimental feature that might increase transfer rate stability and throughput by predicting network congestion. When set to `unset`, the client-specified predictor is used and if the client does not specify a predictor, then none is used. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90.	`unset`, `none`, `alphabeta`, `fd31`, `bezier`, `ets`	`unset`
Outgoing Rate Control Target Queue	The method for calculating the target queue. Static queuing is good for most internet connections, whereas dynamic queuing is good for satellite and other radio connections. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90. When set to `unset`, the client-specified transfer queuing method is used and if the client does not specify a queuing method, then `static` is used.	`unset`, `static`, `dynamic`	`unset`

Controlling Bandwidth Usage with Virtual Links (GUI)

FASP transfers attempt to transfer at the maximum transfer rate available. However, too many simultaneous transfers can overwhelm your storage or leave little bandwidth available for other network activity. To set a bandwidth cap on the total bandwidth used by incoming or outgoing transfer sessions initiated by all users, groups, or sets of specific users, set up a virtual link (Vlink).

About this task

Vlinks are "virtual" bandwidth caps, in that they are not assigned to a specific transfer session, but to all sessions assigned to the same Vlink. The total bandwidth that is used by all incoming or outgoing transfer sessions initiated by users who are assigned to the same Vlink does not exceed the Vlink capacity.

For example, if you want to limit all incoming FASP transfers to 100 Mbps, you can create a Vlink with a 100 Mbps capacity and assign it globally to all incoming transfers. If a user attempts an upload at 50 Mbps but other incoming transfers are already using 75 Mbps, then the transfer rates adjust (based on transfer policy) so that the total does not exceed 100 Mbps.

For another example, if you want to limit to 10 Mbps the total bandwidth that is used by outgoing FASP transfers (downloads) that are initiated by three specific users, create a Vlink with a 10 Mbps capacity and assign it to outgoing transfers for those three users. If the three users are running download sessions that already use 10 Mbps and another download is started by one of the users, the transfer rates of all sessions adjust so that the total bandwidth use by those users remains 10 Mbps. Transfers by other users that are not assigned the Vlink are not affected, except to use available bandwidth when the Vlink capacity is not met.

Procedure

1. Launch the application with administrator privileges.
2. Click **Configuration > Vlinks**.

3. Create a Vlink.

   Click + to add a new Vlink entry; enter a number between 1 and 255 and click **OK**.

4. Configure the Vlink.

   Once the Vlink is created, name it, activate it, and set the bandwidth capacity cap. See the following table for details.

Field	Description	Values	Default
Vlink Name	The Vlink name. This value has no impact on actual bandwidth capping.	text string	blank
On	Select **true** to activate the Vlink; select **false** to deactivate it.	true or false	false
Capacity (Kbps)	Set the virtual bandwidth cap in Kbps. When you apply the Vlink to a transfer (see below), the total bandwidth of all transfers assigned to this vlink is restricted to this value.	positive integer in Kbps	50000

5. Apply a Vlink to users.

   Assign a Vlink to global, group, or user settings. The example below assigns a Vlink to a user's incoming transfer session.

   In the **Configuration** window, click the **Users** tab and select the user to whose transfers you want to apply the Vlink. In the right panel, click the **Bandwidth** tab, select the override box in the **Incoming Vlink ID** row and select the Vlink to apply from the drop-down menu:

6. Prevent users from overriding the Vlink settings.

   If a user requests a minimum rate that exceeds the Vlink and minimum rates are not locked, the transfer can exceed Vlink limits. To prevent this:

   a) Ensure that **Incoming Minimum Rate Default** or **Outgoing Minimum Rate Default** (depending on the direction of the Vlink) is set to zero (the default value).

   b) Select the **Override** box for **Incoming Minimum Rate Lock** or **Outgoing Minimum Rate Lock** (depending on the direction of the Vlink) and select **true**.

Network Configuration

The **Network** configuration options include the network IP, port, and socket buffer settings.

1. Open the application with root privileges.
2. Click **Configuration > Network**.

3. Edit **Global** , **Groups**, and **Users** settings on their **Network** tabs. Select **Override** in the option's row to set an effective value. User settings take precedence over group settings, which take precedence over global settings.

Network Settings Reference

Option	Description	Values	Default
Bind IP Address	Specify an IP address for server-side **ascp** to bind its UDP connection. If a valid IP address is given, **ascp** sends and receives UDP packets only on the interface corresponding to that IP address.	valid IPv4 address	None specified

Option	Description	Values	Default
	Important: The bind address should only be modified (changed to an address other than 127.0.0.1) if you, as the System Administrator, understand the security ramifications of doing so, and have undertaken precautions to secure the SOAP service.		
Bind UDP Port	Prevent the client-side **ascp** process from using the specified UDP port.	integer between 1 and 65535	33001
Disable Packet Batching	Set to `true` to send data packets back-to-back (no sending a batch of packets). This results in smoother data traffic at a cost of higher CPU usage.	`true` or `false`	`false`
Maximum Socket Buffer (bytes)	Set the upper bound of the UDP socket buffer of an **ascp** session below the input value. The default of `0` will cause the Aspera sender to use its default internal buffer size, which may be different for different operating systems.	positive integer	`0`
Minimum Socket Buffer (bytes)	Set the minimum UDP socket buffer size for an ascp session.	positive integer	`0`
RTT auto correction	Set to `true` to enable auto correction of the base (minimum) RTT measurement. This feature is helpful for maintaining accurate transfer rates in hypervisor-based virtual environments.	`true` or `false`	`false`
Reverse path congestion inference	Set to `true` to prevent the transfer speed of a session from being adversely affected by congestion in the reverse (non data-sending) transfer direction. This feature is useful for boosting speed in bi-directional transfers.	`true` or `false`	`true`

File Handling Configuration

The **File Handling** configuration options include file block size, overwrite rules, symbolic link handling, and filtering patterns.

1. Open the application with root privileges.
2. Click **Configuration > File Handling**.
3. Edit **Global** , **Groups**, and **Users** settings on their **File Handling** tabs. Select **Override** in the option's row to set an effective value. User settings take precedence over group settings, which take precedence over global settings.

File Handling Settings Reference

Field	Description	Values	Default
Run File Validation at File Start	Validate files by using the specified method when starting a file transfer (before file transfer starts). For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`uri`, `lua_script`, or none	none

Field	Description	Values	Default
Run File Validation at File Stop	Validate files by using the specified method when file transfer is complete and file is closed. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`uri`, `lua_script`, or none	none
Run File Validation at Session Start	Validate files by using the specified method when a transfer session starts. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`lua_script` or none	none
Run File Validation at Session Stop	Validate files by using the specified method when a transfer session ends. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`lua_script` or none	none
Run File Validation when Crossing File Threshold (Validation Threshold)	Validate files by using the specified method once the transfer session surpasses a set number of kilobytes (threshold). The threshold must be specified by editing `aspera.conf`. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177. **Note:** For threshold validation, the file transfer might complete before the file threshold validation response comes back (because **ascp** doesn't pause file transfers during file threshold validation); therefore, a complete file transfer could happen even with validation failure.	`uri`, `lua_script`, or none	none
Base64-Encoded Lua Action Script	For Lua API validation, the path to the base64-encoded Lua script. This value or "File Path to Lua Action Script" must be defined if any of the following values are set to `lua_script`: Run at File Start, Run at File Stop, Run at Session Start, Run at Session Stop, Run when Crossing File Threshold. If both this option and File Path to Lua Action Script option are defined, this value is ignored. For more information on inline file validation, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	Base64-encoded string	blank
File Path to Lua Action Script	For Lua API validation, the path to the Lua script, or the Base64-encoded Lua script. For detailed information, see “Inline File Validation” on page 111.	Filepath	blank
File Create Mode	Set the file creation mode (permissions). If specified, create files with these permissions (for example, 0755), irrespective of File Create Grant Mask and permissions of the file on the	positive integer (octal)	undefined

Field	Description	Values	Default
	source computer. Only takes effect when the server is a non-Windows receiver.		
File Create Grant Mask	Set the mode for newly created files if File Create Mode is not specified. If specified, file modes will be set to their original modes plus the Grant Mask values. Only takes effect when the server is a non-Windows receiver and when File Create Mode is not specified.	positive integer (octal)	644
Directory Create Mode	Set the directory creation mode (permissions). If specified, create directories with these permissions irrespective of Directory Create Grant Mask and permissions of the directory on the source computer. Only takes effect when the server is a non-Windows receiver.	positive integer (octal)	undefined
Directory Create Grant Mask	Set the mode for newly created directories if Directory Create Mode is not specified. If specified, directory modes will be set to their original modes plus the Grant Mask values. Only takes effect when the server is a non-Windows receiver and when Directory Create Mode is not specified.	positive integer (octal)	755
Read Block Size (bytes)	Set the maximum number of bytes that can be stored within a block as the block is being transferred from the source disk drive to the receiver. The default of zero causes the Aspera sender to use its default internal buffer size, which may vary by operating system. This is a performance-tuning parameter for an Aspera sender (which only takes effect if the *sender* is a server).	positive integer, where 500MB or 524,288,000 bytes is the maximum block size.	0
Write Block Size (bytes)	Set the maximum bytes within a block that an **ascp** receiver can write to disk. The default of zero causes the Aspera receiver to use its default internal buffer size, which may vary by operating system. This is a performance-tuning parameter for an Aspera receiver (which only takes effect if the *receiver* is a server).	positive integer, where 500MB or 524,288,000 bytes is the maximum block size.	0
Number of I/O read threads	Set the number of threads the Aspera sender uses to read file contents from the source disk drive. It takes effect on both client and server, when acting as a sender. The default of zero causes the Aspera sender to use its internal default, which may vary by operating system. This is a performance-tuning parameter for an Aspera sender.	positive integer	0
Number of I/O Write Threads	Set the number of threads the Aspera receiver uses to write the file contents to the destination disk drive. It takes effect on both client and server, when acting as a receiver. The default of zero causes the Aspera receiver to use its internal default, which may vary by operating	positive integer	0

Field	Description	Values	Default
	system. This is a performance-tuning parameter for an Aspera receiver.		
Number of Dir Scanning Threads	Set the number of threads the Aspera sender uses to scan directory contents. It takes effect on both client and server, when acting as a sender. The default of zero causes the Aspera sender to use its internal default. This is a performance-tuning parameter for an Aspera sender.	positive integer	0
Number of Metadata Threads	Set the number of threads the Aspera receiver uses to create directories or 0 byte files. It takes effect on both client and server, when acting as a receiver. The default of zero causes the Aspera receiver to use its internal default, which may vary by operating system. This is a performance-tuning parameter for an Aspera receiver.	positive integer	0
Number of Worker Threads	Set the number of threads the Aspera sender and receiver use to delete files. This is a performance-tuning parameter.	positive integer	0
Sparse File Checking	Set to `true` to enable sparse file checking, which tells the Aspera receiver to avoid writing zero blocks and save disk space. The default of `false` to tell the Aspera receiver to write all the blocks. This is a performance-tuning parameter for an Aspera receiver.	`true` or `false`	`false`
Behavior on Attr Error	Set behavior for when operations attempt to set or change file attributes (such as POSIX ownership, ACLs, or modification time) and fail. Setting to `yes` returns an error and causes the operation to fail. Setting to `no` logs the error and the operation continues	`no` or `yes`	`yes`
Compression Method for File Transfer	Set the compression method to apply to transfers. It applies to both the client and server.	`lz4`, `qlz`, `zlib`, or `none`	`lz4`
Use File Cache	Set to `true` (default) to enable per-file memory caching at the data receiver. File level memory caching improves data write speed on Windows platforms in particular, but uses more memory. This is a performance tuning parameter for an Aspera receiver. Aspera suggests using a file cache on systems that are transferring data at speeds close to the performance of their storage device, and disable it for system with very high concurrency (because memory utilization will grow with the number of concurrent transfers).	`true` or `false`	`true`
Max File Cache Buffer (bytes)	Set the maximum size allocated for per-file memory cache (see Use File Cache) in bytes.	positive integer	`0`

Field	Description	Values	Default
	The default of zero will cause the Aspera receiver to use its internal buffer size, which may be different for different operating systems. This is a performance tuning parameter for an Aspera receiver.		
Resume Suffix	Set the file name extension for temporary metadata files used for resuming incomplete transfers. Each data file in progress will have a corresponding metadata file with the same name plus the resume suffix specified by the receiver. Metadata files in the source of a directory transfer are skipped if they end with the sender's resume suffix.	text string	`.aspx`
Symbolic Link Actions	Set how the server handles symbolic links. For more information about the actions and the interaction between the server configuration and the client request, see "Symbolic Link Handling" on page 150. Combinations of values are logically ORed before use. For example, use `none` alone to mean skip, and shut out other options; when both `follow` and `follow_wide` are set, the latter is recognized.	`none`, `create`, `follow`, `follow_wide`, or any combination of the above delimited by commas	`follow,create`
Preserve Attributes	Set the file creation policy. Set to `none` to not preserve the timestamps of source files. Set to `times` to preserve the timestamp of the source files at destination. **Note:** For Limelight storage, only the preservation of modification time is supported.	`none` or `times`	blank (use the client setting)
Overwrite	Set to `allow` to allow Aspera clients to overwrite existing files on the server, as long as file permissions allow that action. If set to `deny`, clients who upload files to the server cannot overwrite existing files, regardless of file permissions.	`allow` or `deny`	`allow`
File Manifest	Set to `text` to generate a text file "receipt" of all files within each transfer session. Set to `disable` to not create a File Manifest. The file manifest is a file containing a list of everything that was transferred in a given transfer session. The filename of the File Manifest itself is automatically generated based on the transfer session's unique ID. The location where each manifest is written is specified by the File Manifest Path value. If no File Manifest Path is specified, the file will be generated under the destination path at the receiver, and under the first source path at the sender.	`text`, `disable`, or `none`	`none`
File Manifest Path	Specify the location to store manifest files. Can be an absolute path or a path relative to the transfer user's home.	text string	blank

Field	Description	Values	Default
	Note: File manifests can only be stored locally. Thus, if you are using S3, or other non-local storage, you must specify a *local* manifest path.		
File Manifest Suffix	Specify the suffix of the manifest file during file transfer.	text string	`.aspera-inprogress`
Pre-Calculate Job Size	Set to `yes` to enable calculating job size before transferring. Set to `no` to disable calculating job size before transferring. Set to `any` to follow client configurations.	`yes`, `no`, or `any`	`any`
Convert Restricted Windows Characters	To enable the replacement of reserved Windows characters in file and directory names with a non-reserved character, set to the single byte, non-restricted character that will be used for the replacement. Only applies to files written to the local Windows file system; to enable on the peer it must be set on the peer's system.	single-byte, non-restricted character	blank
File Filter Pattern List	Exclude or include files and directories with the specified pattern in the transfer. Add multiple entries for more inclusion/exclusion patterns. To specify an inclusion, start the pattern with '+ ' (+ and a whitespace). To specify an exclusion, start the pattern with '- ' (- and a whitespace). Two symbols can be used in the setting of patterns: • A "**`*`**" (asterisk) represents zero to many characters in a string. For example, `*.tmp` matches `.tmp` and `abcde.tmp`. • A "**`?`**" (question mark) represents a single character. For example, `t?p` matches `tmp` but not `temp`. For details on specifying rules, see “Using Filters to Include and Exclude Files” on page 145. This option applies only when the server is acting as a client. Servers cannot exclude files or directories uploaded or downloaded by remote clients.	text entries	blank
Partial File Name Suffix	Set the filename extension on the destination computer while the file is being transferred. Once the file has been completely transferred, this filename extension is removed. **Note:** This option only takes effect when it is set on the receiver side.	text string	blank
File Checksum Method	Set the type of checksum to calculate for transferred files. The content of transfers can be verified by comparing the checksum value at the destination with the value read at the	`any`, `md5`, `sha1`, `sha256`,	`any`

Field	Description	Values	Default
	source. For more information, see “Reporting Checksums” on page 55.	sha384, or sha512	
Async Log Directory	Set an alternative location for the IBM Aspera Sync log files. If empty, log files go to the default location, or the location specified by the client with `-R`.	filepath	blank
Async Log Level	Set the amount of detail in the Aspera Sync server activity log.	`disable`, `log`, `dbg1`, or `dbg2`	`log`
Async Snapdb Directory	Set an alternative location for the Aspera Sync snapshot database files.	filepath	blank

Configuring Inline File Validation

If an executable file containing malicious code is uploaded to the server, the malicious code can subsequently be executed by an external product that integrates with an Aspera product. Inline file validation is a feature that enables file content to be validated while the file is in transit, as well as when the transfer is complete. The validation check can be made with a Lua script, or with a REST call to an external URL. The mode of validation used (URL or Lua) and the timing of the check are set in `aspera.conf`.

About this task

When URI inline file validation is enabled, the transfer is not reported as complete until the validation completes. An alternative to inline file validation, out-of-transfer file validation, completes the transfer and then validates the file, and can be substantially faster. For more information, see “Out-of-Transfer File Validation” on page 109.

Note: Inline file validation is not applied to transfers that fall back to HTTP. If all transfers require validation, use out-of-transfer validation.

Note: Lua scripting is supported for many uses, including inline file validation. For detailed information, see “Automated Execution of Lua Scripts with Transfer Events” on page 177.

Procedure

1. For Lua script validation, specify the path to it or the base-64 encoded script itself.

   Go to **Configuration > File Handling** for a specific user and set either **Base64-Encoded Lua Action Script** or **File Path to Lua Action Script**, depending on if your script is base64 encoded.
2. For URI validation, configure the REST service and set the URL.

   Note: The code examples provided here are for an admin using a Java servlet deployed on an Apache web server, but this process is generalizable to other programming languages and other servers.

   a) Open `web.xml` and edit the `<servlet>` and `<servlet_mapping>` sections to provide the necessary information for validation.

   The `<servlet-name>` (URL handler) value is also configured in `aspera.conf` (in the next step) and any custom code (such as file filtering, see “Inline File Validation with URI” on page 113).

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://
xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

    <servlet>
```

```
        <servlet-name>SimpleValidator</servlet-name>
        <servlet-class>aspera.validation.SimpleValidator</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>SimpleValidator</servlet-name>
        <url-pattern>/SimpleValidator/validation/files</url-pattern>
    </servlet-mapping>
</web-app>
```

b) Set the URL in `aspera.conf`.

```
# asconfigurator -x "set_user_data;user_name,username;validation_uri,url"
```

Where *url* is the server's IP address and port, and the servlet name (URL handler) found in `web.xml`. This adds the path to the `<transfer>` section of `aspera.conf`. For example:

```
<transfer>
<validation_uri>http://127.0.0.1:8080/SimpleValidator</validation_uri>
</transfer>
```

3. Schedule the validation.

   Go to **Configuration > File handling** and select **uri** or **lua_script** to schedule that type of validation at the following events:

   - **Run File Validation at File Start**
   - **Run File Validation at File Stop**
   - **Run File Validation at Session Start** (URL validation is not supported)
   - **Run File Validation at Session Stop** (URL validation is not supported)
   - **Run File Validation When Crossing File Threshold**

   You can set a Lua script validation to run at one event and a URI validation to run at another, but you can define only one Lua script or URL. The default setting for all events is **none**.

4. If you schedule validation at a file size threshold, set the threshold.

   This setting cannot be done in the GUI; run the following command:

   ```
   # asconfigurator -x "set_user_data;user_name,username;validation_threshold_kb,size"
   ```

5. Configure multi-threaded validation.

   By default, inline validation is set to use 5 threads.

   If the number of validation threads is not set to 1, then multiple threads may perform different types of validations for different (or the same) files at the same time. In such a situation, the response of a `validation_file_stop` at the end of a file download might come before the response of a `validation_threshold` for the same file.

   To set the number of validation threads, run the following command:

   ```
   # asconfigurator -x "set_user_data;user_name,username;validation_threads,number"
   ```

Results

For more information about the configuration parameters, see “aspera.conf - Transfer Configuration” on page 70 (defining values in `aspera.conf`)

For more information on the output of your inline validation, see “Inline File Validation with URI” on page 113 or “Inline File Validation with Lua Script” on page 115.

Configuring 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.

Filters can be specified on the **ascp** command line and in `aspera.conf`. Ascp applies filtering rules that are set in `aspera.conf` *before* it applies rules on the command line.

Set Filtering Rules in the GUI

1. Click **Configuration > File Handling**.
2. Scroll down to **File Filter Pattern List**.
3. Select **Override** then click ... to open the filter **Define** dialog. If rules were added earlier, either through the GUI or through `aspera.conf`, they will appear in the window.
4. Click the + button to add a new filtering rule, or click the − button to delete a rule that you've selected.

Rule Syntax

A rule consists of a "+" or "-" sign (indicating whether to include or exclude), followed by a space character, followed by a pattern. A pattern can be a file or directory name, or a set of names expressed with UNIX *glob* patterns.

Note: Do not confuse the GUI line-add and line-delete buttons in the GUI: + and −, with the include/exclude characters "+" or "-" that are part of rule syntax. The purpose of each is different and they are unrelated.

Basic usage

- Filtering rules are applied to the transfer list in the order from top to bottom.
- Filtering is a process of exclusion, and include rules override exclude rules that follow them. Include rules cannot add back files that are excluded by a preceding exclude rule.
- Include rules must be followed by at least one exclude rule, otherwise all files are transferred because none are excluded. To exclude all unmatched files, add two final rules: "- *" and "- .*".
- Filtering operates only on the set of files and directories in the transfer list. An include rule cannot add files or directories that are not already part of the transfer list.

Example	Transfer Result
- *rule*	Transfer all files and directories except those with names that match *rule*.

+ *rule*	Transfer all files and directories because none are excluded.
	To transfer only the files and directories with names that match *rule* use:
	`+ rule`
	`- *`
	`- .*`

+ *rule1* - *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*.
- *rule1* + *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'`. Additional filters can be set for

Example	Transfer Result

	or on the command line (“Using Filters to Include and Exclude Files” on page 145).
	To transfer only the files and directories with names that do not match *rule1* but do match *rule2* use:
	`- rule1`
	`+ rule2`
	`- *`
	`- .*`

Filtering Rule Application

Filtering order

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

1. The first file (or directory) in the transfer list is compared to the pattern of the first rule.
2. If the file matches the pattern, Ascp includes it or excludes it 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.
3. If the file does not match, it is compared to 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 set of rules:

```
+ file2
- file[0-9]
```

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

1. `file1` is compared with the first rule (`+ file2`) and does not match so filtering continues.
2. `file1` is compared with the second rule (`- 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.

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
[!...]	[!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 Filtering Rules

You can use this procedure to test your filtering rules.

1. On your computer, create a small set of directories and files that generally matches a file set you typically transfer. Since filenames are all that matter, the size of the files can be small.
2. Place the file set in an accessible location, for example `/tmp/src`.
3. Upload the file set to your server. For information about setting up a connection, see "Testing a Locally Initiated Transfer" on page 18.
4. Create a destination directory on your computer, for example `/tmp/dest`.
5. You can now download your files from the server to `/tmp/dest` and test your filtering rules. For example:
6. Compare the list of files transferred to the list of your original files.

Example Filter Rules

The example rules below are based on downloading a directory AAA from your server to `/tmp/dest` on your computer:

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 results:

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

(1) Transfer everything except files and directories starting with "**.**":

```
+ *
- 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

```
- 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:

```
+ wxy*
- 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:

```
+ */wxy*
- 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 "**.**":

```
+ */wxy*
- */wxy/**
- 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:

```
+ /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 "`.`":

```
+ AAA/abc/wxy/**
- 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
```

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 HSTS or HSTE 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
```

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 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.

Transfer Server Configuration

HSTS uses asperacentral to handle transfer requests from Aspera clients. You can configure server properties and behavior in the **Transfer Server** options, including specifying the address, enabling persistent storage, and controlling how to handle empty files.

1. Open HSTS with root privileges.
2. Click **Configuration > Global > Transfer Server**.
3. Edit settings on the **Transfer Server** tab. Select **Override** in the option's row to set an effective value.

Transfer Server Settings Reference

Setting	Description	Values	Default
Address	The network interface address on which the transfer server listens. The default value of 127.0.0.1 enables the transfer server to accept transfer requests from the local computer. If you set the address to 0.0.0.0, the transfer server can accept requests on all network interfaces. Alternatively, a specific network interface address may be specified.	Valid IPv4 address	127.0.0.1
Port	The port on which the transfer server accepts transfer requests.	Positive integer 1 - 65535	40001
Persistent Storage	Enable to retain data that is stored in the database between reboots of asperacentral.	Enable or Disable	Enable
Persistent Storage Path	The location in which to store data between reboots of asperacentral. If the path is a directory, then a file is created with the default name `central-store.db`. Otherwise, the file is named as specified in the path.	Valid system path	If the application is installed in the default location, then the path is the following: `/opt/aspera/var/`

Setting	Description	Values	Default
Maximum Age (seconds)	Maximum allowable age (in seconds) of data to be retained in the database.	Positive integer	86400
Exit Central on Storage Error	The behavior of the asperacentral server if a database write error occurs.	Ignore or Exit	Ignore
Compact Database on Startup	Enable or disable compacting (vacuuming) the database when the transfer server starts.	Enable or Disable	Enable
Files Per Session	The maximum number of files that can be retained for persistent storage.	Positive integer	1000
Ignore Empty Files	Set to **true** to block the logging of zero-byte files.	true or false	true
Ignore No-transfer Files	Set to **true** to block the logging of files that were not transferred because they exist at the destination.	true or false	true
Post-Transfer Validation Timeout	How many seconds to wait for a post-transfer validator to update the status of a file before the file is released from the validator and its status is changed back to "to_be_validated". This allows a file to be validated by a different validator if the first validator stops working. For more information, see “Out-of-Transfer File Validation” on page 109.	Positive integer	300

Set up Users and Groups

Aspera clients connect to HSTS by authenticating as a system user who is configured in the application. The user can also belong to a group that is configured in the application. Users and groups can be set up by running asconfigurator commands or directly editing the configuration file, `aspera.conf`.

Setting Up Transfer Users

HSTS uses system accounts to authenticate connections from Aspera clients. The system users must be added and configured as Aspera transfer users before clients can browse the server file system or run FASP transfers to and from the server. When creating transfer users, you can also specify user-specific settings, such as transfer bandwidth, docroot, and file handling. User configuration is an important part of securing your server. For a complete description, see “Aspera Ecosystem Security Best Practices” on page 346.

About this task

Important Configuration Notes:

- Some Aspera features require a docroot in URI format or require a file restriction instead of a docroot. For more information, see “Docroot vs. File Restriction” on page 345.
- If users connect to the server by providing IBM Aspera Shares credentials or by providing Node API credentials that are associated with the transfer user, changes to a user's configuration, such as their docroot, are not applied to the user until asperanoded is restarted. For instructions, see “Restarting Aspera Services” on page 344.

To configure a system user account as an Aspera transfer user:

Procedure

1. Create default (global) transfer settings.

   To set default values to prohibit transfers in and out, set the encryption key, and set the default docroot for all users, run the following commands (if not already set):

   ```
   # asconfigurator -x "set_node_data;authorization_transfer_in_value,deny"
   # asconfigurator -x "set_node_data;authorization_transfer_out_value,deny"
   # asconfigurator -x "set_node_data;token_encryption_key,token_key"
   # asconfigurator -x "set_node_data;absolute,docroot"
   ```

   For server security, Aspera recommends the following settings:

   - Deny transfers by default, then enable transfers for individual users as required (described in a later step).
   - Set the token encryption key to a string of at least 20 random characters.
   - Set a default docroot to an empty folder or a part of the file system specific to each user.

   If there is a pattern in the docroot of each user, for example, *username*, you can use a substitutional string. This way you assign independent docroot to each user without setting a docroot for each user individually

Substitutional String	Definition	Example
`$(name)`	system user's name	`/sandbox/$(name)`
`$(home)`	system user's home directory	`$(home)/Documents`

2. For server security, Aspera recommends restricting users' read, write, and browse permissions.

   Users are given read, write, and browse permissions to their docroot by default. For increased security, change the global default to deny these permissions:

   ```
   # asconfigurator -x "set_node_data;read_allowed,false;write_allowed,false;dir_allowed,false"
   ```

   Run the following commands to enable permissions per user, as required:

   ```
   # asconfigurator -x "set_user_data;user_name,username;read_allowed,true"
   # asconfigurator -x "set_user_data;user_name,username;write_allowed,true"
   # asconfigurator -x "set_user_data;user_name,username;dir_allowed,true"
   ```

3. If you provided an Aspera license during installation (rather than an entitlement), ensure that the transfer user has read permissions on the Aspera license file (`aspera-license`) so that they can run transfers.

   The license file is found in: `/opt/aspera/etc/`

4. Restrict user permissions with **`aspshell`**.

   By default, all system users can establish a FASP connection and are only restricted by file permissions. Restrict the user's file operations by assigning them to use **`aspshell`**, which permits only the following operations:

   - Running Aspera uploads and downloads to or from this computer.
   - Establishing connections in the application.
   - Browsing, listing, creating, renaming, or deleting contents.

   These instructions explain one way to change a user account or active directory user account so that it uses the **`aspshell`**; there may be other ways to do so on your system.

   Run the following command to change the user login shell to **`aspshell`**:

   ```
   # sudo usermod -s /bin/aspshell username
   ```

Confirm that the user's shell updated by running the following command and looking for `/bin/aspshell` at the end of the output:

```
# grep username /etc/passwd
username:x:501:501:...:/home/username:/bin/aspshell
```

Note: If you use OpenSSH, sssd, and Active Directory for authentication: To make `aspshell` the default shell for all domain users, first set up a local account for server administration because this change affects all domain users. Then open `/etc/sssd/sssd.conf` and change `default_shell` from `/bin/bash` to `/bin/aspshell`.

5. Configure user-specific transfer settings.

   Besides the default (global) transfer settings, you can create user-specific and group-specific transfer settings. The user-specific settings have the highest priority, overriding both group and global settings. For more information, see "Configuration Precedence" on page 23.

   To set user-specific values to authorize transfers in and out, docroot, and target rate, run the following commands:

   ```
   # asconfigurator -x "set_user_data;user_name,username;authorization_transfer_in_value,allow"
   # asconfigurator -x "set_user_data;user_name,username;authorization_transfer_out_value,allow"
   # asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
   # asconfigurator -x
   "set_user_data;user_name,username;transfer_in_bandwidth_flow_target_rate_default,rate"
   # asconfigurator -x
   "set_user_data;user_name,username;transfer_out_bandwidth_flow_target_rate_default,rate"
   ```

   For more information about other user settings, see "aspera.conf - Authorization Configuration" on page 68, "aspera.conf - Transfer Configuration" on page 70, and "aspera.conf - File System Configuration" on page 91.

6. Verify the configuration.

   If you modify `aspera.conf` by editing the text, use the following command to verify the XML form and values:

   ```
   # /opt/aspera/bin/asuserdata -v
   ```

7. Restart asperanoded and asperacentral to activate your changes.

   Run the following commands to restart asperanoded:

   ```
   # systemctl restart asperanoded
   ```

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

   ```
   # service asperanoded restart
   ```

   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
   ```

Setting Up Transfer Groups

Transfer settings can be applied to your system's user groups. If users within a group do not have individual transfer settings, then the group's transfer settings are applied. HSTS doesn't create user

groups on the operating system for you, so you must ensure that the groups exist before adding them to your Aspera product.

Procedure

1. Determine the user groups to add to HSTS.

   Ensure that you have an existing user group on your operating system, or create a new user group. Please refer to your operating system's documentation for information on creating user groups. HSTS reads group information from the following file:

   ```
   /etc/group
   ```

2. Add the user group to your Aspera transfer product

   When a transfer group is specified, it overwrites global settings and applies group configuration to corresponding users. To add group-specific transfer settings, you can use **asconfigurator** commands with the following syntax:

   ```
   # asconfigurator -x "set_group_data;group_name,groupname;parameter,value"
   ```

   For more information on available settings, see “User, Group and Default Configurations” on page 328 and the references in the table below.

Category	Description
“Configuration Precedence” on page 23	When a user is a member of multiple groups, the precedence setting can be used to determine priority.
“aspera.conf - Authorization Configuration” on page 68	Connection permissions, token key, and encryption requirements.
“aspera.conf - Transfer Configuration” on page 70	Incoming and outgoing transfer bandwidth and policy settings.
“aspera.conf - File System Configuration” on page 91	Docroot, file and directory creation, access permissions, block sizes, and so on.

   You can also manually edit `aspera.conf` with a text editor.

   `/opt/aspera/etc/aspera.conf`

   Add the following section to `aspera.conf`:

```
<?xml version='1.0' encoding='UTF-8'?>
<CONF version="2">
   <aaa>
      <realms>
         <realm>
            <users>
               ... <!-- user-specific settings -->
            </users>
            <groups>
               <group> <!-- Each group tag contains a group's profile. -->
                  <name>aspgroup</name> <!-- The group name. -->
                  <precedence>0</precedence> <!-- Group precedence. -->
                  <authorization>...</authorization> <!-- Authorization settings. -->
                  <transfer>...</transfer> <!-- Transfer settings. -->
                  <file_system>...</file_system> <!-- File System settings. -->
               </group>
               <group>
                  ... <!-- Another group's settings-->
               </group>
            </groups>
         </realm>
      <realms>
   </aaa>
   ...
</CONF>
```

3. Configure the group's transfer settings.

Settings	Description
“Configuration Precedence” on page 23	When a user is a member of multiple groups, the precedence setting can be used to determine priority.
“aspera.conf - Authorization Configuration” on page 68	Connection permissions, token key, and encryption requirements.
“aspera.conf - Transfer Configuration” on page 70	Incoming and outgoing transfer bandwidth and policy settings.
“aspera.conf - File System Configuration” on page 91	Docroot, file and directory creation, access permissions, block sizes, and so on.

You can also manually edit `aspera.conf` with a text editor.

`/opt/aspera/etc/aspera.conf`

Add the following section to `aspera.conf`:

```
<?xml version='1.0' encoding='UTF-8'?>
<CONF version="2">
   <aaa>
      <realms>
         <realm>
            <users>
               ... <!-- user-specific settings -->
            </users>
            <groups>
               <group> <!-- Each group tag contains a group's profile. -->
                  <name>aspgroup</name> <!-- The group name. -->
                  <precedence>0</precedence> <!-- Group precedence. -->
                  <authorization>...</authorization> <!-- Authorization settings. -->
                  <transfer>...</transfer> <!-- Transfer settings. -->
                  <file_system>...</file_system> <!-- File System settings. -->
               </group>
               <group>
                  ... <!-- Another group's settings-->
               </group>
            </groups>
         </realm>
      <realms>
   </aaa>
   ...
</CONF>
```

4. Verify your configuration.

When you have finished updating the group's settings in `aspera.conf`, use the following command to verify it (in this example, verify the group asp-group's settings):

```
# /opt/aspera/bin/asuserdata -g asp-group
```

5. Restart asperanoded and asperacentral to activate your changes.

Run the following commands to restart asperanoded:

```
# systemctl restart asperanoded
```

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

```
# service asperanoded restart
```

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
```

Configuration Precedence

HSTS applies configuration settings in this order: 1) user settings, 2) group settings (and if a user belongs to more than one group, precedence can be set for each group), 3) global settings, 4) default settings. User settings have the highest priority and default the lowest.

For example, the table below shows the setting values that are applied to user `aspera_user_1` in **bold** when that user is also a member of several groups and global settings are configured. In this example, `aspera_user_1` is a member of both the **admin** and **xfer** groups. The **admin** group's precedence setting is 0, which supersedes the **xfer** group's setting of 1:

Options	"aspera_user_1" User Settings	"admin" Group Settings	"xfer" Group Settings	Global Settings	Default Settings
Target rate	**5M**	10M	15M	40M	45M
Min rate	n/a	**2M**	8M	3M	0
Policy	n/a	n/a	**Low**	Fair	Fair
Docroot	n/a	n/a	n/a	**/pod/$(name)**	n/a
Encryption	n/a	n/a	n/a	n/a	**any**

Configuring Precedence of Groups

You can configure a group's precedence in `aspera.conf` by running the following **asconfigurator** command:

```
# asconfigurator -x "set_group_data;group_name,group_name;precedence,value"
```

Note: A group's precedence setting must be greater than or equal to 0, where 0 is the highest precedence level.

This adds a <group> section to `aspera.conf` like the one below. In this example, group "admin" has higher precedence than group "xfer".

```
<groups>
   <group>
      <name>admin</name>
      <precedence>0</precedence>
      ...
   </group>
   <group>
      <name>xfer</name>
      <precedence>1</precedence>
      ...
   </group>
</groups>
```

You can also edit `aspera.conf` manually by opening it with administrative privileges:

`/opt/aspera/etc/aspera.conf`

In the file, locate the entry for each group, add the <precedence> option, and assign a precedence value as shown in the example above. After editing the file, validate the XML form and option values by running the following command:

```
# /opt/aspera/bin/asuserdata -v
```

Setting Up a User's Public Key on the Server

Public key authentication is an alternative to password authentication, providing a more secure authentication method that allows users to avoid entering or storing a password, or sending it over the network. An Aspera client generates a key pair (a public key and a private key) on the client computer and provides the public key to the administrator of the remote Aspera transfer server. The server administrator sets up the client user's public key as described in the following steps.

About this task

For information on how to create public keys, see “Creating SSH Keys ” on page 152.

Procedure

1. Obtain the client user's public key.

   The client user should send you a secure email with the public key pasted in the message body or attached as a text file.
2. Install the public key in the user account on the server.
   a) In the home directory of the account that the client will use to access the server, create a directory called `.ssh` if it doesn't already exist.
   b) Save the key file as `authorized_keys` in `.ssh`. If `authorized_keys` already exists, append the key file to it.

      For example,

      ```
      # mkdir /home/aspera_user_1/.ssh
      # cat /tmp/id_rsa.pub > /home/aspera_user_1/.ssh/authorized_keys
      ```

      Where:

      - `aspera_user_1` is the server user account.
      - `/tmp/id_rsa.pub` is where you saved the public key sent by the user.
      - `/home/aspera_user_1/.ssh/authorized_keys` is the file that contains the public key.

   c) Configure permissions on the key.

      Make the system user (in this example, user `aspera_user_1`) the owner of key directory and key file, allow access by the `aspera_user_1` group, and set permissions:

      ```
      # chown -R aspera_user_1:aspera_user_1  /home/aspera_user_1/.ssh
      # chmod 700 /home/aspera_user_1
      # chmod 700 /home/aspera_user_1/.ssh
      # chmod 600 /home/aspera_user_1/.ssh/authorized_keys
      ```

Testing a User-Initiated Remote Transfer

Once you have configured an Aspera transfer user on HSTS, test that an Aspera client can successfully connect to HSTS and upload a file.

About this task

Prerequisites:

- **Client:** Install an Aspera client application, such as the freely available IBM Aspera Desktop Client or IBM Aspera Command-Line Interface, on the client computer.
- **Server:** HSTS must have at least one Aspera transfer user (a system user account that is configured to authenticate Aspera transfers) configured on it.

If any of the following connection tests fail, see “Clients Cannot Establish Connection” on page 341.

Procedure

1. On the client, test that you can reach the IP address of your server.

   Run the **ping** command:

   ```
   # ping 10.0.0.2
   PING 10.0.0.2 (10.0.0.2): 56 data bytes
   64 bytes from 10.0.0.2: icmp_seq=0 ttl=64 time=8.432 ms
   64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=7.121 ms
   64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=5.116 ms
   64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=4.421 ms
   64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.050 ms
   ...
   ```

   In the example above, the address of HSTS is 10.0.0.2 and the output shows successful responses from the host.

   If the output returns "Destination host unreachable," check the firewall configuration of the server.

2. On the client, try a transfer to HSTS by using **ascp**.

   Run the following command on your client machine:

   ```
   # ascp -P 33001 --mode=send --policy=fair -l 10000 -m 1000 source_path
   username@ip_address:destination
   ```

   For example: (where `aspera_user_1` is the example transfer user):

   ```
   # ascp -P 33001 --mode=send --policy=fair -l 10000 -m 1000 /client-dir/files
   aspera_user_1@10.0.0.2:/dir
   ```

   This command specifies the following values for the transfer:

Item	Argument	Example Value
TCP Port Set the TCP port to start the transfer session.	`-P` *`port`*	`-P 33001`
Transfer Direction Specify if the server is the destination or source.	`--mode=`*`direction`*	`--mode=send`
Transfer Policy Specify how to share bandwidth with other network users.	`--policy=`*`policy`*	`--policy=fair`
Maximum Transfer Rate Set the maximum transfer rate, in Kbps.	`-l` *`rate`*	`-l 10000` Maximum transfer rate = 10 Mbps
Minimum Transfer Rate Set the minimum transfer rate, in Kbps.	`-m` *`rate`*	`-l 1000` Minimum transfer rate = 1 Mbps
File or Directory to Upload Set the path relative to your current directory.	*source*	`/client-dir/files`
Transfer User	*username*	`aspera_user_1`
Host Address	*ip_address*	`10.0.0.2`
Destination Folder	*destination*	`/dir`

Item	Argument	Example Value
Set the destination path relative to the transfer user's docroot.		In this example, the files are transferred to the "dir" folder in the docroot of aspera_user_1.

Configure the Server from the Command Line

The following references describe the server settings that can be configured for HSTS by using the command line or directly editing the HSTS configuration file, `aspera.conf`.

aspera.conf - Websocket Configuration

The `<server>` section of `aspera.conf` can be used to configure the server to use the Websocket protocol instead of SSH or HTTPS. The ascp client uses HTTPS for Websocket only. However, the Websocket server can be configured to use HTTP as long as a proxy is being used to terminate the HTTPS server endpoint.

About this task

Configuration methods: These instructions describe how to manually modify `aspera.conf`. You can also add and edit these parameters using **`asconfigurator`** commands. For more information on using **`asconfigurator`**, see "User, Group and Default Configurations" on page 328 and run the following command to retrieve a complete default `aspera.conf` that includes the **`asconfigurator`** syntax for each setting:

```
# /opt/aspera/bin/asuserdata -+
```

Procedure

1. Open `aspera.conf` from the following location:

   `/opt/aspera/etc/aspera.conf`
2. Add or locate the `<server>` section, as in the following example:

   ```
   <server>
       <enable_http> true</enable_http>
       <enable_https> true</enable_https>
       <wss_port>9093</wss_port>
       <wss_enabled>true</wss_enabled>
   </server>
   ```
3. Edit settings as needed.

   Websocket Settings Reference

Field	Description	Values
Enable HTTP	Enable HTTP	`true` or `false`
Enable HTTPS	Enable HTTPS	`true` or `false`
WSS Port	Websocket port	`9093`
WSS Enabled	Enable Websocket	`true` or `false`
4. Save and validate `aspera.conf`.

   Run the following command to confirm that the XML is correctly formatted and the parameter settings are valid:

   ```
   # /opt/aspera/bin/asuserdata -v
   ```

aspera.conf - Authorization Configuration

The settings in the <authorization> section of aspera.conf include transfer permissions and token configuration. Tokens are used by Aspera web applications to authorize transfers between Aspera clients and servers.

About this task

Note: For security, Aspera recommends denying incoming and outgoing transfers globally, then allowing transfers by individual users, as needed. For a compilation of server security best practices, see "Aspera Ecosystem Security Best Practices" on page 346.

Configuration methods: These instructions describe how to manually modify aspera.conf. You can also add and edit these parameters using **asconfigurator** commands. For more information on using **asconfigurator**, see "User, Group and Default Configurations" on page 328 and run the following command to retrieve a complete default aspera.conf that includes the **asconfigurator** syntax for each setting:

```
# /opt/aspera/bin/asuserdata -+
```

Procedure

1. Open aspera.conf from the following location:

   /opt/aspera/etc/aspera.conf

2. Add or locate the <authorization> section, as in the following example:

```
<authorization>
   <transfer>
      <in>
         <value>allow</value>       <!-- Incoming Transfer -->
         <external_provider>
            <url>...</url>   <!-- Incoming External Provider URL -->
            <soap>...</soap> <!-- Incoming External Provider SOAP Action -->
         </external_provider>
      </in>
      <out>
         <value>allow</value>       <!-- Outgoing Transfer -->
         <external_provider>
            <url>...</url>   <!-- Outgoing External Provider URL -->
            <soap>...</soap> <!-- Outgoing External Provider SOAP Action -->
         </external_provider>
      </out>
   </transfer>
   <token>
     <encryption_type>aes-128</encryption_type> <!-- Token Encryption Cipher -->
     <encryption_key> </encryption_key>         <!-- Token Encryption Key -->
     <filename_hash> </filename_hash>           <!-- Token Filename Hash -->
     <life_seconds>86400</life_seconds>         <!-- Token Life (seconds) -->
   </token>
</authorization>
```

3. Edit settings as needed.

   Authorization Settings Reference

Field	Description	Values	Default
Incoming Transfers	To enable users to transfer to this computer, leave the default setting of allow. Set to deny to prevent transfers to this computer. Set to token to allow only transfers initiated with valid tokens to this computer. Token-based transfers are typically used by web applications such as IBM Aspera Faspex and IBM Aspera Shares and require a Token Encryption Key.	allow, deny, or token	allow

Field	Description	Values	Default
Incoming External Provider URL	Set the URL of the external authorization provider for incoming transfers. The default empty setting disables external authorization. Aspera servers can be configured to check with an external authorization provider. This SOAP authorization mechanism can be useful to organizations requiring custom authorization rules. Requires a value for Incoming External Provider SOAP Action.	HTTP URL	blank
Incoming External Provider SOAP Action	The SOAP action required by the external authorization provider for incoming transfers. Required if Incoming External Provider URL is set.	text string	blank
Outgoing Transfers	To enable users to transfer from this computer, leave the default setting of `allow`. Set to `deny` to prevent transfers from this computer. Set to `token` to allow only transfers initiated with valid tokens from this computer. Token-based transfers are typically used by web applications such as Faspex and require a Token Encryption Key.	`allow`, `deny`, or `token`	`allow`
Outgoing External Provider URL	Set the URL of the external authorization provider for outgoing transfers. The default empty setting disables external authorization. HSTS can be configured to check with an external authorization provider. This SOAP authorization mechanism can be useful to organizations requiring custom authorization rules. Requires a value for Outgoing External Provider Soap Action.	HTTP URL	blank
Outgoing External Provider Soap Action	The SOAP action required by the external authorization provider for outgoing transfers. Required if Outgoing External Provider URL is set.	text string	blank
Token Encryption Cipher	Set the cipher used to generate encrypted transfer tokens.	`aes-128`, `aes-192`, or `aes-256`	`aes-128`
Token Encryption Key	Set the secret text phrase that is used to authorize those transfers configured to require token. Aspera recommends setting a token encryption key of at least 20 random characters. For more information, see “Require Token Authorization: Set from the Command Line” on page 310.	text string	blank

Field	Description	Values	Default
Token Filename Hash	Set the algorithm with which filenames inside transfer tokens should be hashed. Use MD5 for backward compatibility.	sha1, md5, or sha-256	sha-256
Token Life (seconds)	Set the token expiration for users of web-based transfer applications.	positive integer	86400 (24 hrs)

4. Save and validate `aspera.conf`.

   Run the following command to confirm that the XML is correctly formatted and the parameter settings are valid:

```
# /opt/aspera/bin/asuserdata -v
```

aspera.conf - Transfer Configuration

The settings in the <transfer> section of `aspera.conf` include: bandwidth control, transfer protocol options, content encryption requirements, encryption-at-rest, and inline validation.

About this task

Configuration methods: These instructions describe how to manually modify `aspera.conf`. You can also add and edit these parameters using **asconfigurator** commands. For more information on using **asconfigurator**, see "User, Group and Default Configurations" on page 328 and run the following command to retrieve a complete default `aspera.conf` that includes the **asconfigurator** syntax for each setting:

```
# /opt/aspera/bin/asuserdata -+
```

Procedure

1. Open `aspera.conf` from the following location:

   `/opt/aspera/etc/aspera.conf`

2. Add or locate the <transfer/> section, as in the following example:

```
<transfer>
 <in>
    <bandwidth>
       <aggregate>
          <trunk_id>Disabled</trunk_id>        <!-- Incoming VLink ID -->
       </aggregate>
       <flow>
          <target_rate>
             <cap></cap>                  <!-- Incoming Target Rate Cap -->
             <default>10000</default>     <!-- Incoming Target Rate Default -->
             <lock>false</lock>           <!-- Incoming Target Rate Lock -->
          </target_rate>
          <min_rate>
             <cap></cap>                  <!-- Incoming Minimum Rate Cap -->
             <default>0</default>          <!-- Incoming Minimum Rate Default -->
             <lock>false</lock>           <!-- Incoming Minimum Rate Lock -->
          </min_rate>
          <policy>
             <allowed>any</allowed        <!-- Incoming Policy Allowed -->
             <default>fair</default>      <!-- Incoming Policy Default -->
             <lock>false</lock>           <!-- Incoming Policy Lock -->
          </policy>
          <priority>
             <cap></cap>                  <!-- Incoming Priority Allowed -->
             <default>normal</default>    <!-- Incoming Priority Default -->
             <lock>false</lock>           <!-- Incoming Priority Lock -->
          </priority>
          <network_rc>
             <module>delay</module>             <!-- Incoming Rate Control Module -->
             <tcp_friendly>false</tcp_friendly>    <!-- Incoming TCP Friendly Mode -->
             <predictor>unset</predictor>       <!-- Incoming Traffic RTT Predictor -->
```

```
            <target_queue>unset</target_queue> <!-- Incoming Rate Control Target Queue -->
         </network_rc>
      </flow>
   </bandwidth>
</in>
<out>
   <bandwidth>
      <aggregate>
         <trunk_id>Disabled</trunk_id>   <!-- Outgoing VLink ID -->
      </aggregate>
      <flow>
         <target_rate>
            <cap>Unlimited</cap>         <!-- Outgoing Target Rate Cap -->
            <default>10000</default>     <!-- Outgoing Target Rate Default -->
            <lock>false</lock>           <!-- Outgoing Target Rate Lock -->
         </target_rate>
         <min_rate>
            <cap>Unlimited</cap>         <!-- Outgoing Minimum Rate Cap -->
            <default>0</default>         <!-- Outgoing Minimum Rate Default -->
            <lock>false</lock>           <!-- Outgoing Minimum Rate Lock -->
         </min_rate>
         <policy>
            <allowed>any</allowed>       <!-- Outgoing Policy Allowed -->
            <default>fair</default>      <!-- Outgoing Policy Default -->
            <lock>false</lock>           <!-- Outgoing Policy Lock -->
         </policy>
         <priority>
            <cap>high</cap>              <!-- Outgoing Priority Allowed -->
            <default>normal</default>    <!-- Outgoing Priority Default -->
            <lock>false</lock>           <!-- Outgoing Priority Lock -->
         </priority>
         <network_rc>
            <module>delay</module>             <!-- Outgoing Rate Control Module -->
            <tcp_friendly>false</tcp_friendly> <!-- Outgoing TCP Friendly Mode -->
            <predictor>unset</predictor>       <!-- Outgoing Traffic RTT Predictor -->
            <target_queue>unset</target_queue> <!-- Outgoing Rate Control Target Queue -->
         </network_rc>
      </flow>
   </bandwidth>
</out>
<encryption>
    <allowed_cipher>any</allowed_cipher>  <!-- Encryption Allowed -->
    <fips_mode>false</fips_mode>          <!-- Transfer in FIPS-140-2-certified encryption
mode -->
    <strict_allowed_cipher>false</strict_allowed_cipher>
                          <!-- Allow transfer when client lacks GCM   -->
    <content_protection_required>false
    </content_protection_required>
                                          <!-- Content Protection Required -->
    <content_protection_secret></content_protection_secret>
                                          <!-- Content Protection Secret -->
    <content_protection_strong_pass_required>false
    </content_protection_strong_pass_required>
                                          <!-- Strong Password Required for Content
Protection -->
</encryption>
<protocol_options>
    <bind_ip_address></bind_ip_address>         <!-- Bind IP Address -->
    <bind_udp_port>33001</bind_udp_port>        <!-- Bind UDP Port -->
    <disable_batching>false</disable_batching>  <!-- Disable Packet Batching -->
    <batch_size>0</batch_size>                  <!-- Batch Size -->
    <datagram_size>0</datagram_size>            <!-- Datagram Size -->
    <max_sock_buffer>0</max_sock_buffer>        <!-- Maximum Socket Buffer (bytes)-->
    <min_sock_buffer>0</min_sock_buffer>        <!-- Minimum Socket Buffer (bytes)-->
    <rtt_autocorrect>true</rtt_autocorrect>     <!-- RTT auto correction -->
    <rtt_reverse_infer>true</rtt_reverse_infer> <!-- Reverse path congestion inference -->
    <chunk_size>0</chunk_size>                  <!-- Chunk Size -->
</protocol_options>
<chunker_max_mem></chunker_max_mem>             <!-- Limit memory sender will use -->
<validation_file_start>none</validation_file_start>
                                               <!-- Validation File Start -->
<validation_file_stop>none</validation_file_stop>
                                               <!-- Validation File Stop -->
<validation_session_start>none</validation_session_start>
                                               <!-- Validation Session Start -->
<validation_session_stop>none</validation_session_stop>
                                               <!-- Validation Session Stop -->
<validation_threshold>none</validation_threshold>
                                               <!-- Validation Threshold -->
<validation_uri>AS_NULL</validation_uri>
                                               <!-- Validation URI -->
<validation_threshold_kb>0</validation_threshold_kb>
```

```
                                                    <!-- Validation Threshold KB -->
 <validation_threads>5</validation_threads>    <!-- Validation Threads -->
 <validation_lua_script_base64></validation_lua_script_base64>
                                                     <!-- Validation Lua Script Base64 -->
 <validation_lua_script_path></validation_lua_script_path>
                                                    <!-- Validation Lua Script Path -->
</transfer>
```

For information about configuration for monitoring transfers with Lua scripts with the various `validation` tags, see "Configuration for Lua Script Execution" on page 177.

3. Edit settings as needed.

Transfer Settings Reference

Field	Description	Values	Default
Incoming Vlink ID	The ID of the vlink to apply to incoming transfers. Vlinks are a way to define aggregate transfer policies. For more information, see "Controlling Bandwidth Usage with Virtual Links (Command Line)" on page 87.	Vlink IDs	`Undefined` (Disabled)
Incoming Target Rate Cap (Kbps)	The maximum target rate for incoming transfers, in kilobits per second. No transfer session can exceed this rate at any time. If the client requests an initial rate greater than the target rate cap, the transfer proceeds at the target rate cap. The default setting of `unlimited` applies no target rate cap.	positive integer	unlimited
Incoming Target Rate Default (Kbps)	The default initial rate for incoming transfers, in kilobits per second. If allowed ("Incoming Target Rate Lock" is set to `false`), clients can modify this rate in real time. This setting is not relevant to transfers with a `fixed` bandwidth policy.	positive integer	`10000`
Incoming Target Rate Lock	Lock the target rate of incoming transfers to the default value (set to `true`). Set to `false` to allow users to adjust the transfer rate of an incoming transfer up to the "Incoming Target Rate Cap".	`true` or `false`	`false`
Incoming Minimum Rate Cap (Kbps)	The highest minimum rate that an incoming transfer can request, in kilobits per second. Client minimum rate requests that exceed the minimum rate cap are ignored. The default value of `unlimited` applies no cap to the minimum rate. **Important:** Aspera strongly recommends setting the minimum rate cap to zero. Transfers do not slow below the client's requested minimum rate unless the minimum rate is capped on the server. If the client-requested minimum rate exceeds	positive integer or `unlimited`	`unlimited`

Field	Description	Values	Default
	network or storage capacity, this can decrease transfer performance and cause problems on the target storage.		
Incoming Minimum Rate Default (Kbps)	The default initial minimum rate for incoming transfers, in kilobits per second. If allowed ("Incoming Minimum Rate Lock" is set to `false`), clients can modify the minimum rate in real time, up to the "Incoming Minimum Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy.	positive integer	`0`
Incoming Minimum Rate Lock	Lock the minimum rate of incoming transfers to the default value (set to `true`). Set to `false` to allow users to adjust the minimum transfer rate up to the "Incoming Minimum Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy. **Important:** Aspera strongly recommends setting a lock on minimum rate to prevent transfers from using minimum rates that can overwhelm network or storage capacity, decrease transfer performance, and cause problems on the target storage.	`true` or `false`	`false`
Incoming Bandwidth Policy Allowed	The bandwidth policies that incoming transfers can use. Aspera transfers can use high, fair, low, or fixed bandwidth policies to determine bandwidth allocation among transfers. • `any` - The server does not deny any transfer based on policy setting. **Note:** Setting to `any` allows clients to request a `fixed` bandwidth policy. If the client also requests a high minimum transfer rate and that is not capped by the server, the transfer rate can exceed network or storage capacity. This can decrease transfer performance and cause problems on the target storage. To avoid these problems, set the allowed policy to `fair`. • `high` - Transfers that use `high`, `fair`, or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected.	`high`, `fair`, `low`, or `any`	`any`

Field	Description	Values	Default
	• `fair` - Transfers that use `fair` or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected. • `low` - Only transfers that use a `low` bandwidth policy are allowed. All others are rejected.		
Incoming Bandwidth Policy Default	The default bandwidth policy for incoming transfers. Clients can override the default policy if they specify a policy allowed by the server (see "Incoming Bandwidth Policy Allowed") and if "Incoming Bandwidth Policy Lock" is set to `false`. • `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.	`high`, `fair`, `low`, `fixed`	`fair`
Incoming Bandwidth Policy Lock	Lock the bandwidth policy of incoming transfer sessions to the default value	`true` or `false`	`false`

Field	Description	Values	Default
	(set to `true`). Set to `false` to allow users to adjust the bandwidth policy.		
Incoming Priority Allowed	The highest priority the client can request. Use the value 0 to unset this option; 1 to allow high priority, 2 to enforce normal priority.	`0`, `1`, or `2`	`1`
Incoming Priority Default	The initial priority setting. Use the value 0 to unset this option, 1 to allow high priority; 2 to enforce normal priority	`0`, `1`, or `2`	`2`
Incoming Priority Lock	To disallow your clients change the priority, set the value to true	`true` or `false`	`false`
Incoming Rate Control Module	Set how the transmission rate should be managed relative to instantaneous network bandwidth availability. Aspera recommends that this option be changed only by advanced users. When the client does not specify a configuration, the server configuration is used. When the client specifies a value other than `delay` and the client is the receiver, then the client configuration overrides the server configuration. Values: • **delay:** The baseline rate control module used by Aspera transfers. • **delay-odp:** A queue-scaling controller for overdrive protection. • **delay-adv:** An advanced rate controller. • **delay-laq:** A loss-adjusted queueing (LAQ) rate controller. **Note:** The LAQ module is an experimental rate control module that is designed to solve issues with target rate overdrive, high concurrency (when many FASP sessions run at the same time), and shallow buffers (limited packet queuing capability of a router). When LAQ is set, then it uses the FD31 RTT predictor unless a different RTT predictor is explicitly set. To set a rate control module for outgoing traffic, set it from the command line (“aspera.conf - Transfer Configuration” on page 70).	`delay`, `delay-odp`, `delay-adv`, or `delay-laq`	`delay`

Field	Description	Values	Default
TCP Friendly (for *incoming* rate control)	This setting is meant for advanced users to turn TCP-friendly mode on or off (which is only applied at the local "receiver" side when the transfer policy is set to `fair`). It should only be used with special instructions for debugging. When enabled ("`true`"), incoming FASP transfers are allowed to maintain relative fair bandwidth share with a TCP flow under congestion.	`true` or `false`	`false`
Incoming Traffic RTT Predictor	The type of predictor to use to compensate for feedback delay when measuring RTT. An experimental feature that might increase transfer rate stability and throughput by predicting network congestion. When set to `unset`, the client-specified predictor is used and if the client does not specify a predictor, then none is used. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90.	`unset`, none, `alphabeta`, `fd31`, `bezier`, `ets`	`unset`
Incoming Rate Control Target Queue	The method for calculating the target queue. Static queuing is good for most internet connections, whereas dynamic queuing is good for satellite and other radio connections. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90. When set to `unset`, the client-specified transfer queuing method is used and if the client does not specify a queuing method, then `static` is used.	`unset`, `static`, `dynamic`	`unset`
Outgoing Vlink ID	The ID of the vlink to apply to outgoing transfers. Vlinks are a way to define aggregate transfer policies. For more information, see "Controlling Bandwidth Usage with Virtual Links (Command Line)" on page 87.	Vlink ID	`Undefined` (Disabled)
Outgoing Target Rate Cap (Kbps)	The maximum target rate for outgoing transfers, in kilobits per second. No transfer session can exceed this rate at any time. If the client requests an initial rate greater than the target rate cap, the transfer proceeds at the target rate cap. The default setting of `unlimited` applies no target rate cap.	positive integer	`unlimited`
Outgoing Target Rate Default (Kbps)	The default initial rate for outgoing transfers, in kilobits per second. If allowed ("Outgoing Target Rate Lock"	positive integer	`10000`

Field	Description	Values	Default
	is set to `false`), clients can modify this rate in real time up to the "Outgoing Target Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy.		
Outgoing Target Rate Lock	Lock the target rate of outgoing transfers to the default value (set to `true`). Set to `false` to allow users to adjust the transfer rate of an outgoing transfer.	`true` or `false`	`false`
Outgoing Minimum Rate Cap (Kbps)	The highest minimum rate that an outgoing transfer can request, in kilobits per second. Client minimum rate requests that exceed the minimum rate cap are ignored. The default value of `unlimited` applies no cap to the minimum rate. **Important:** Aspera strongly recommends setting the minimum rate cap to zero. Transfers do not slow below the client's requested minimum rate unless the minimum rate is capped on the server. If the client-requested minimum rate exceeds network or storage capacity, this can decrease transfer performance and cause problems on the target storage.	positive integer	`unlimited`
Outgoing Minimum Rate Default	The default initial minimum rate for outgoing transfers, in kilobits per second. If allowed ("Outgoing Minimum Rate Lock" is set to `false`), clients can modify the minimum rate in real time up to the "Outgoing Minimum Rate Cap". This setting is not relevant to transfers with a `fixed` bandwidth policy.	positive integer	`0`
Outgoing Minimum Rate Lock	Lock the minimum rate of outgoing transfers to the default value (set to `true`). Set to `false` to allow users to adjust the minimum transfer rate. This setting is not relevant to transfers with a `fixed` bandwidth policy. **Important:** Aspera strongly recommends setting a lock on minimum rate to prevent transfers from using minimum rates that can overwhelm network or storage capacity, decrease transfer performance, and cause problems on the target storage.	`true` or `false`	`false`

Field	Description	Values	Default
Outgoing Bandwidth Policy Allowed	The bandwidth policies that outgoing transfers can use. Aspera transfers can use high, fair, low, or fixed bandwidth policies to determine bandwidth allocation among transfers. • `any` - The server does not deny any transfer based on policy setting. **Note:** Setting to `any` allows clients to request a `fixed` bandwidth policy. If the client also requests a high minimum transfer rate and that is not capped by the server, the transfer rate can exceed network or storage capacity. This can decrease transfer performance and cause problems on the target storage. To avoid these problems, set the allowed policy to `fair`. • `high` - Transfers that use `high`, `fair`, or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected. • `fair` - Transfers that use `fair` or `low` bandwidth policies are allowed. Transfers that request `fixed` bandwidth policy are rejected. • `low` - Only transfers that use a `low` bandwidth policy are allowed. All others are rejected.	`high`, `fair`, `low`, or `any`	`any`
Outgoing Bandwidth Policy Default	The default bandwidth policy for outgoing transfers. Clients can override the default policy if they specify a policy allowed by the server (see "Outgoing Bandwidth Policy Allowed") and if "Outgoing Bandwidth Policy Lock" is set to `false`. • `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	`high`, `fair`, `low`, `fixed`	`fair`

Field	Description	Values	Default
	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.		
Outgoing Bandwidth Policy Lock	Lock the bandwidth policy of outgoing transfer sessions to the default value (set to `true`). Set to `false` to allow users to adjust the bandwidth policy.	`true` or `false`	`false`
Outgoing Priority Allowed	The highest priority your client can request. Use the value 0 to unset this option; 1 to allow high priority, 2 to enforce normal priority.	`0`, 1, or 2	1
Outgoing Priority Default	The initial priority setting. Use the value 0 to unset this option, 1 to allow high priority; 2 to enforce normal priority.	`0`, 1, or 2	2
Outgoing Priority Lock	To prevent your clients from changing the priority, set the value to `true`.	`true` or `false`	false
Outgoing Rate Control Module	Set how the transmission rate should be managed relative to instantaneous network bandwidth availability. Aspera recommends that this option be changed only by advanced users. When the client does not specify a configuration, the server configuration is used. When the client specifies a value other than `delay` and the client is the receiver, then the client configuration overrides the server configuration. Values:	`delay`, `delay-odp`, `delay-adv`, or `delay-laq`	`delay`

Field	Description	Values	Default
	• **delay:** The baseline rate control module used by Aspera transfers. • **delay-odp:** A queue-scaling controller for overdrive protection. • **delay-adv:** An advanced rate controller. • **delay-laq:** A loss-adjusted queueing (LAQ) rate controller. **Note:** The LAQ module is an experimental rate control module that is designed to solve issues with target rate overdrive, high concurrency (when many FASP sessions run at the same time), and shallow buffers (limited packet queuing capability of a router). When LAQ is set, then it uses the FD31 RTT predictor unless a different RTT predictor is explicitly set.		
TCP Friendly (for *outgoing* rate control)	This setting is meant for advanced users to turn TCP-friendly mode on or off (which is only applied at the local "receiver" side when the transfer policy is set to `fair`). It should only be used with special instructions for debugging. When enabled ("`true`"), outgoing FASP transfers are allowed to maintain relative fair bandwidth share with a TCP flow under congestion.	`true` or `false`	false
Outgoing Traffic RTT Predictor	The type of predictor to use to compensate for feedback delay when measuring RTT. An experimental feature that might increase transfer rate stability and throughput by predicting network congestion. When set to `unset`, the client-specified predictor is used and if the client does not specify a predictor, then none is used. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90.	`unset`, none, `alphabeta`, `fd31`, `bezier`, `ets`	`unset`
Outgoing Rate Control Target Queue	The method for calculating the target queue. Static queuing is good for most internet connections, whereas dynamic queuing is good for satellite and other radio connections. For more information, see "Increasing Transfer Performance by Using an RTT Predictor" on page 90. When set to `unset`, the client-specified transfer queuing method is used and if the	`unset`, `static`, `dynamic`	`unset`

Field	Description	Values	Default
	client does not specify a queuing method, then `static` is used.		
Content Protection Required	Set to `true` to require that uploaded content be encrypted by the client (enforce client-side encryption-at-rest). For more information, see "Client-Side Encryption-at-Rest (EAR)" on page 156. **Important:** When a transfer falls back to HTTP or HTTPS, content protection is no longer supported. If HTTP fallback occurs while downloading, then–despite entering a passphrase–the file remains encrypted. If HTTP fallback occurs during upload, then–despite entering a passphrase–the files are not encrypted.	`true` or `false`	`false`
Strong Password Required for Content Encryption	Set to `true` to require that the password for content encryption (client-side encryption at rest) includes at least 6 characters, of which at least 1 is non-alphanumeric, at least 1 is a letter, and at least 1 is a digit.	`true` or `false`	`false`
Content Protection Secret	Enable server-side encryption-at-rest (EAR) by setting the passphrase. Files uploaded to the server are encrypted while stored there and are decrypted when they are downloaded. For more information, see "Server-Side Encryption-at-Rest (EAR)" on page 103.	passphrase	(none)
Encryption Allowed	Set the transfer encryption allowed by this computer. Aspera strongly recommends that you require transfer encryption. 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. **Note:** To ensure client compatibility when requiring encryption, use a cipher with the form `aes-XXX`, which is supported by all clients and servers. Requiring GCM causes the server to reject transfers from clients that are	`any`, `none`, `aes-128`, `aes-192`, `aes-256`, `aes-128-cfb`, `aes-192-cfb`, `aes-256-cfb`, `aes-128-gcm`, `aes-192-gcm`, or `aes-256-gcm`	`any`

Field	Description	Values	Default
	running a version of Ascp 3.8 or older, unless <strict_allowed_cipher> is set to false. When a client requests a shorter cipher key than is configured on the server (or in an access key that authorizes the transfer), the transfer is automatically upgraded to the server setting. For more information about how the server and client negotiate the transfer cipher, see the description of -c in "Ascp Command Reference" on page 121 and "Ascp4 Command Reference" on page 162. Values: • any - allow transfers that use any encryption cipher or none. • none - require unencrypted transfers (not recommended). • aes-128, aes-192, or aes-256 - allow transfers that use an encryption cipher key that is as long or longer than the setting. These settings use the CFB or GCM mode depending on the client version and cipher requested. Supports all client versions. • aes-128-cfb, aes-192-cfb, or aes-256-cfb - require that transfers use the CFB encryption mode and a cipher key that is as long or longer than the setting. Supports all client versions. • aes-128-gcm, aes-192-gcm, or aes-256-gcm - require that transfers use the GCM encryption mode introduced in version 3.9.0 and a cipher that is as long or longer than the setting.		
Do encrypted transfers in FIPS-140-2-certified encryption mode	Set to true for **ascp** to use a FIPS 140-2-certified encryption module. When enabled, transfer start is delayed while the FIPS module is verified. When you run **ascp** in FIPS mode (that is, <fips_enabled> is set to true in aspera.conf), and you use passphrase-protected SSH keys, you must use keys generated by running **ssh-keygen** in a FIPS-enabled system, or convert existing keys to a	true or false	false

Field	Description	Values	Default
	FIPS-compatible format using a command such as the following: `openssl pkcs8 -topk8 -v2 aes128 -in id_rsa -out new-id_rsa` **Important:** When set to `true`, all ciphers and hash algorithms that are not FIPS compliant will abort transfers.		
Bind IP Address	Specify an IP address for server-side **ascp** to bind its UDP connection. If a valid IP address is given, **ascp** sends and receives UDP packets only on the interface corresponding to that IP address. **Important:** The bind address should only be modified (changed to an address other than 127.0.0.1) if you, as the System Administrator, understand the security ramifications of doing so, and have undertaken precautions to secure the SOAP service.	valid IPv4 address	None specified
Bind UDP Port	Prevent the client-side **ascp** process from using the specified UDP port.	integer between 1 and 65535	33001
Disable Packet Batching	Set to `true` to send data packets back-to-back (no sending a batch of packets). This results in smoother data traffic at a cost of higher CPU usage.	`true` or `false`	`false`
Batch Size	When set to "0" (default), the system uses a pre-computed batch size. Set this to "1" for high concurrency servers (senders) in order to reduce CPU utilization in aggregate.	Integer	0
Datagram Size	Sets the datagram size on the server side. If size is set with both **-Z** (client side) and <datagram_size> (server side), the <datagram_size> setting is used. In cases where the client-side is pre-3.3, datagram size is determined by the **-Z** setting, regardless of the server-side setting for <datagram_size>. In such cases, if there is no **-Z** setting, datagram size is based on the discovered MTU and the server logs the message "LOG Peer client doesn't support alternative datagram size".	Integer	1492
Maximum Socket Buffer (bytes)	Set the upper bound of the UDP socket buffer of an **ascp** session below the	positive integer	0

Field	Description	Values	Default
	input value. The default of `0` will cause the Aspera sender to use its default internal buffer size, which may be different for different operating systems.		
Minimum Socket Buffer (bytes)	Set the minimum UDP socket buffer size for an ascp session.	positive integer	`0`
RTT auto correction	Set to `true` to enable auto correction of the base (minimum) RTT measurement. This feature is helpful for maintaining accurate transfer rates in hypervisor-based virtual environments.	`true` or `false`	`false`
Reverse path congestion inference	Set to `true` to prevent the transfer speed of a session from being adversely affected by congestion in the reverse (non data-sending) transfer direction. This feature is useful for boosting speed in bi-directional transfers.	`true` or `false`	`true`
Chunk Size	For multi-session transfers with object storage, the chunk size must be equal to or greater than the object storage part size. For more information, see “Multi-Session Transfers” on page 140.	positive integer	`0`
Limit memory sender will use	Use of `<chunker_max_mem>` is necessary when you have a high transfer rate, a significantly lossy or slow network between the sender and receiver, are sending large files (in the Gigabyte range), and the sending host does not have a generous amount of RAM to spare. The `<chunker_max_mem>` element limits the amount of memory (defined in bytes) that the sender will use to hold on to data that has yet to be acknowledged by the receiver. This means that the sender will temporarily stop reading data that it will send as for as long as that limit is reached.	positive integer	none
Run File Validation at File Start	Validate files by using the specified method when starting a file transfer (before file transfer starts). For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`uri`, `lua_script`, or `none`	none

Field	Description	Values	Default
Run File Validation at File Stop	Validate files by using the specified method when file transfer is complete and file is closed. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`uri`, `lua_script`, or `none`	`none`
Run File Validation at Session Start	Validate files by using the specified method when a transfer session starts. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`lua_script` or `none`	`none`
Run File Validation at Session Stop	Validate files by using the specified method when a transfer session ends. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	`lua_script` or `none`	`none`
Run File Validation when Crossing File Threshold (Validation Threshold)	Validate files by using the specified method once the transfer session surpasses a set number of kilobytes (threshold). The threshold must be specified by editing `aspera.conf`. For more information, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177. **Note:** For threshold validation, the file transfer might complete before the file threshold validation response comes back (because **ascp** doesn't pause file transfers during file threshold validation); therefore, a complete file transfer could happen even with validation failure.	`uri`, `lua_script`, or `none`	`none`
Validation Threshold KB	Validate files once the download size exceeds the threshold value. Since threshold validation can only be triggered periodically (every second in the worst case), the file must be large enough to trigger this validation. The Validation Threshold option must also be specified (`uri` or `lua`) if this option is to be recognized by the system. If Validation Threshold is also enabled, and this value is not specified (or set	Positive integer	`0`

Field	Description	Values	Default
	to 0), the **ascp** session will exit with an error.		
Validation Threads	Enable multiple validations to occur in parallel validator threads. If the number of validation threads is not set to 1, then multiple threads may perform different types of validations for different (or the same) files at the same time. In such a situation, the response of a `validation_file_stop` at the end of a file download might come before the response of a `validation_threshold` for the same file.	Positive integer	5
Validation URI	Use the specified external URL for validation calls. When this parameter is defined, at least two validations, validation_file_start and validation_file_stop will happen for every file. The entry should define a URL, port, and URL handler for validation. For example, `http://127.0.0.1:8080/SimpleValidator` This value must be defined if any of the following values are set to `uri`: • `validation_file_start` • `validation_file_stop` • `validation_ session_start` • `validation_session_stop` • `validation_threshold`	URL	none
Base64-Encoded Lua Action Script	For Lua API validation, the path to the base64-encoded Lua script. This value or "File Path to Lua Action Script" must be defined if any of the following values are set to `lua_script`: Run at File Start, Run at File Stop, Run at Session Start, Run at Session Stop, Run when Crossing File Threshold. If both this option and File Path to Lua Action Script option are defined, this value is ignored. For more information on inline file validation, see “Inline File Validation” on page 111 and “Automated Execution of Lua Scripts with Transfer Events” on page 177.	Base64-encoded string	blank

Field	Description	Values	Default
File Path to Lua Action Script	For Lua API validation, the path to the Lua script, or the Base64-encoded Lua script. For detailed information, see “Inline File Validation” on page 111.	Filepath	blank

4. Save and validate `aspera.conf`.

   Run the following command to confirm that the XML is correctly formatted and the parameter settings are valid:

```
# /opt/aspera/bin/asuserdata -v
```

Controlling Bandwidth Usage with Virtual Links (Command Line)

FASP transfers attempt to transfer at the maximum transfer rate available. However, too many simultaneous transfers can overwhelm your storage or leave little bandwidth available for other network activity. To set a bandwidth cap on the total bandwidth used by incoming or outgoing transfer sessions initiated by all users, groups, or sets of specific users, set up a virtual link (Vlink).

About this task

Vlinks are "virtual" bandwidth caps, in that they are not assigned to a specific transfer session, but to all sessions assigned to the same Vlink. The total bandwidth that is used by all incoming or outgoing transfer sessions initiated by users who are assigned to the same Vlink does not exceed the Vlink capacity.

For example, if you want to limit all incoming FASP transfers to 100 Mbps, you can create a Vlink with a 100 Mbps capacity and assign it globally to all incoming transfers. If a user attempts an upload at 50 Mbps but other incoming transfers are already using 75 Mbps, then the transfer rates adjust (based on transfer policy) so that the total does not exceed 100 Mbps.

For another example, if you want to limit to 10 Mbps the total bandwidth that is used by outgoing FASP transfers (downloads) that are initiated by three specific users, create a Vlink with a 10 Mbps capacity and assign it to outgoing transfers for those three users. If the three users are running download sessions that already use 10 Mbps and another download is started by one of the users, the transfer rates of all sessions adjust so that the total bandwidth use by those users remains 10 Mbps. Transfers by other users that are not assigned the Vlink are not affected, except to use available bandwidth when the Vlink capacity is not met.

Procedure

1. Create a Vlink.

   To create a Vlink, run the following command as administrator:

```
# asconfigurator -x "set_trunk_data;id,vlink_id;trunk_capacity,bandwidth;trunk_on,true"
```

   You can also specify a multicast port and time-to-live, among other settings. To see a complete list of parameters with their corresponding **`asconfigurator`** commands, run the following command:

```
# /opt/aspera/bin/asuserdata -+
```

   The following table describes several parameters that are frequently used:

Tag	Description	Values	Default
Vlink ID	The Vlink ID. Sessions assigned with the same trunk ID share the same bandwidth cap.	positive integer between 1 and 255.	N/A

Tag	Description	Values	Default
Vlink Name	The Vlink name. This value has no impact on actual bandwidth capping.	text string	blank
Capacity	This value reflects the virtual bandwidth cap in Kbps. When applying this Vlink to a transfer (e.g. Default outgoing), the transfer's bandwidth will be restricted by this value.	positive integer in Kbps	50000
On	Set to **true** to activate this Vlink; set to **false** to deactivate it.	true/false	false
Multicast Port	This sets the UDP port through which virtual link sends and receives multicast communication messages. Sessions sharing the same virtual bandwidth cap needs to have the same port number. To avoid port conflicts, it is recommended to use the default UDP port 55001. Do NOT set the port number to the same one used by FASP data transfer (33001). **Important:** If you have a local firewall on your server (for example, Windows firewall, Linux iptables, or Mac ipfw), you will need to allow the Vlink UDP port (55001, by default) for multicast traffic.	positive integer between 1 and 65535	55001
Multicast TTL	This sets the Time-to-Live (TTL) field in the IP header for Vlink multicast packets.	positive integer between 1 and 255	blank

For example, to create a Vlink with an ID of 108, named "50Mbps cap", with a capacity of 50 Mbps (50000 kbps), run the following command:

```
# asconfigurator -x "set_trunk_data;id,108;trunk_name,50Mbps
cap;trunk_capacity,50000;trunk_on,true"
```

This creates the following text in `aspera.conf`:

```
<CONF version="2">
  ...
  <trunks>
    <trunk>
      <id>108</id>                                    <!-- Vlink ID -->
      <name>50Mbps cap</name>                         <!-- Vlink Name -->
      <capacity>
        <schedule format="ranges">50000</schedule>    <!-- Capacity -->
      </capacity>
      <on>true</on>                                   <!-- On -->
    </trunk>
  </trunks>
</CONF>
```

The capacity of the Vlink is set within a `<schedule>` tag because the capacity can be scheduled as one value during a specified time period, and a default value at all other times. For more information on this configuration, see the knowledge base article *Specifying a time varying schedule for a Vlink* at https://www.ibm.com/mysupport.

To edit `aspera.conf` manually, rather than running **`asconfigurator`** commands, open the file with write permissions from the following location:

`/opt/aspera/etc/aspera.conf`

Validate the `aspera.conf` file using the asuserdata utility:

```
# /opt/aspera/bin/asuserdata -v
```

2. Apply the Vlink.

   Assign a Vlink to global, group, or user settings for transfers in or out. Use the following syntax, updating the direction (in or out) depending on your needs:

```
# asconfigurator -x "set_node_data;transfer_in_bandwidth_aggregate_trunk_id,id"
# asconfigurator -x
"set_group_data;group_name,groupname;transfer_out_bandwidth_aggregate_trunk_id,id"
# asconfigurator -x
"set_user_data;user_name,username;transfer_out_bandwidth_aggregate_trunk_id,id"
```

   For example, to set Vlink 108 as the default for transfers out and set Vlink 109 to the user `aspera_user_1` for transfers out, run the following commands:

```
# asconfigurator -x "set_node_data;transfer_out_bandwidth_aggregate_trunk_id,108"
# asconfigurator -x
"set_user_data;user_name,aspera_user_1;transfer_out_bandwidth_aggregate_trunk_id,109"
```

   These commands add the following lines to `aspera.conf`:

```
<CONF version="2">
  ...
  <default>
    <transfer>
      <out>
        <bandwidth><aggregate>
          <trunk_id>108</trunk_id> <!-- Vlink #108 for the default outgoing sessions. -->
        </aggregate></bandwidth>
      </out>
      <in>
       ...
      </in>
    </transfer>
  </default>
  <aaa><realms><realm>
    <users>
      <user>
        <name>aspera_user_1</name>
        <transfer>
          <out>
            <bandwidth><aggregate>
              <trunk_id>109</trunk_id> <!-- Vlink #109 to the user aspera_user_1's outgoing
sessions. -->
            </aggregate></bandwidth>
          </out>
          <in>
           ...
          </in>
        </transfer>
      </user>
    </users>
  </realm></realms></aaa>
</CONF>
```

3. Prevent users from overriding the Vlink settings.

   If a user requests a high minimum rate and minimum rates are not locked, the transfer can exceed Vlink limits. To prevent this:

   a) Set the default incoming or outgoing minimum rate to zero (zero is the default) by running the appropriate command:

```
# asconfigurator -x "set_node_data;transfer_in_bandwidth_flow_min_rate_default,0"
# asconfigurator -x "set_node_data;transfer_out_bandwidth_flow_min_rate_default,0"
```

   b) Lock the minimum default transfer rate for select users or globally. The following commands lock minimum incoming and outgoing transfer rates for all users:

```
# asconfigurator -x "set_node_data;transfer_in_bandwidth_flow_min_rate_lock,true"
# asconfigurator -x "set_node_data;transfer_out_bandwidth_flow_min_rate_lock,true"
```

Global Bandwidth Settings (Command Line)

Global bandwidth usage by incoming and outgoing transfers can be configured from the command line by creating Vlink(s) that is applied to all users.

In the following example, Vlink 108 is used to limit the upload bandwidth (outgoing transfers) to 88 Mbps (88000 Kbps) and Vlink 109 is used to limit the download bandwidth (incoming transfers) to 99 Mbps (99000 Kbps).

```
# asconfigurator -x "set_trunk_data;id,108;trunk_capacity,88000;trunk_on,true"
# asconfigurator -x "set_trunk_data;id,109;trunk_capacity,99000;trunk_on,true"
# asconfigurator -x "set_node_data;transfer_in_bandwidth_aggregate_trunk_id,108"
# asconfigurator -x "set_node_data;transfer_out_bandwidth_aggregate_trunk_id,109"
```

The commands create the following lines in `aspera.conf`.

```
<?xml version='1.0' encoding='UTF-8'?>
<CONF version="2">
  ...
  <trunks>
    <trunk>          <!-- Create a Vlink with 88000 Kbps bandwidth cap. -->
      <id>108</id>   <!-- ID: 108 -->
      <capacity>
        <schedule format="ranges">88000</schedule>
      </capacity>
      <on>true</on>
    </trunk>
    <trunk>          <!-- Create a Vlink with 99000 Kbps bandwidth cap. -->
      <id>109</id>   <!-- ID: 109 -->
      <capacity>
        <schedule format="ranges">99000</schedule>
      </capacity>
      <on>true</on>
    </trunk>
  </trunks>

  <default>   <!-- Global settings.-->
    <transfer>
      <out>   <!-- Use Vlink ID: 108 for global outgoing bandwidth. -->
        <bandwidth><aggregate><trunk_id>108</trunk_id></aggregate></bandwidth>
      </out>
      <in>    <!-- Use Vlink ID: 109 for global incoming bandwidth. -->
        <bandwidth><aggregate><trunk_id>109</trunk_id></aggregate></bandwidth>
      </in>
    </transfer>
  </default>
</CONF>
```

The capacity of the Vlink is set within a `<schedule>` tag because the capacity can be scheduled as one value during a specified time period, and a default value at all other times. For more information on this configuration, see the knowledge base article *Specifying a time varying schedule for a Vlink* at https://www.ibm.com/mysupport.

To edit `aspera.conf` manually, rather than running **`asconfigurator`** commands, open the file with write permissions from the following location:

`/opt/aspera/etc/aspera.conf`

Validate the `aspera.conf` file using the asuserdata utility:

```
# /opt/aspera/bin/asuserdata -v
```

Increasing Transfer Performance by Using an RTT Predictor

FASP transfers use delay-based congestion control to dynamically adjust the transfer rate in response to network congestion, as measured by round-trip time (RTT). As a result, FASP transfer stability is sensitive to feedback delay; increases in feedback delay decrease FASP transfer stability and throughput. Transfer performance can be improved by using two experimental configuration options, an RTT predictor and dynamic target queuing.

RTT Predictor

An RTT predictor predicts future feedback delay to decrease transfer rate oscillation and maximize data transfer under high network congestion conditions. Four RTT predictors are available:

- **alphabeta:** A linear prediction that is based on a local trend.
- **fd31:** A linear prediction that is based on a 3-points-backwards difference method.
- **bezier:** A quadratic Bezier extrapolation.
- **ets:** An error-trend-seasonality model.

Based on internal testing, fd31 is considered the most effective and robust, but other RTT predictors might perform better depending on your specific network conditions.

To set a predictor for incoming (transfer_in) or outgoing (transfer_out) transfers, run the following command:

```
# asconfigurator -x "set_node_data;transfer_{in|out}_bandwidth_flow_network_rc_predictor,
{alphabeta|bezier|ets|fd31}"
```

You can also set the value to none to force no predictor, or `unset` to use the client-specified predictor. If the client does not specify a predictor and the server is set to `unset`, then no predictor is used.

The fd31 and bezier predictors do not have a bounded asymptotic limit, which can destabilize the RTT prediction under conditions of high congestion and large buffer size for the transfer link. The prediction range can be restricted by setting <`predictor_limit_range`> in `aspera.conf`.

Dynamic Target Queuing

Target queuing affects the stability of data transfer to the target. By default, Aspera FASP transfers use static target queuing, in which the target queue is set as a piecewise function of the target rate. On noisy networks, such as satellite and other radio communication, the congestion signal can be distorted at the physical or data link layer, and this noise can overwhelm the congestion signal. Static target queuing has only a limited ability to adjust to this noise, decreasing transfer performance.

Dynamic target queueing is an experimental method to improve transfer speed and stability over noisy networks. When dynamic target queuing is enabled, the rate control module estimates the noise level and adjusts the target queue accordingly.

To enable dynamic target queuing for incoming (transfer_in) or outgoing (transfer_out) transfers, run the following command:

```
# asconfigurator -x "set_node_data;transfer_{in|
out}_bandwidth_flow_network_rc_target_queue,dynamic"
```

Command line options override server settings. If no predictor is specified on the client command line, in the client's `aspera.conf`, or in the server's `aspera.conf`, then no predictor is used for the transfer.

aspera.conf - File System Configuration

The settings in the <`file_system`> section of `aspera.conf` include the docroot, file permissions, file handling, filters, checksum reporting, and growing files. The absolute path, or docroot, is the area of the file system that is accessible to an Aspera transfer user. The default empty value allows access to the entire file system. You can set one global docroot and then further restrict access to the file system by group or individual user.

About this task

Important Configuration Notes:

- The default server configuration gives users full access to the server's file system with read, write, and browse privileges. Aspera strongly recommends setting a global docroot that is an empty folder and

setting global file permissions to **false**. For a compilation of server security best practices, see "Aspera Ecosystem Security Best Practices" on page 346.

- Some Aspera features require a docroot in URI format or require a file restriction instead of a docroot. For more information, see "Docroot vs. File Restriction" on page 345.

Configuration methods: These instructions describe how to manually modify `aspera.conf`. You can also add and edit these parameters using **`asconfigurator`** commands. For more information on using **`asconfigurator`**, see "User, Group and Default Configurations" on page 328 and run the following command to retrieve a complete default `aspera.conf` that includes the **`asconfigurator`** syntax for each setting:

```
# /opt/aspera/bin/asuserdata -+
```

Procedure

1. Open `aspera.conf` from the following location:

   `/opt/aspera/etc/aspera.conf`

2. Add or locate the `<file_system />` section, as in the following example.

```
<file_system>
   <access>
      <paths>
         <path>
            <absolute peer_ip="ip_address">/path/$(name)</absolute>
                                                    <!-- Absolute Path (conditional) -->
            <absolute>/path/$(name)</absolute>         <!-- Absolute Path -->
<!-- Growing files handling can also be specified within the <absolute> section;
                for detailed informataion, see the table below  -->
            <restrictions>
                <restriction></restriction>       <!-- File Restriction 1 -->
                <restriction></restriction>       <!-- File Restriction 2 -->
            </restrictions>
            <read_allowed>true</read_allowed>          <!-- Read Allowed -->
            <write_allowed>true</write_allowed>        <!-- Write Allowed -->
            <dir_allowed>true</dir_allowed>            <!-- Browse Allowed -->
         </path>
      </paths>
   </access>
   <read_block_size>0</read_block_size>                <!-- Read Block Size -->
   <write_block_size>0</write_block_size>              <!-- Write Block Size -->
   <read_threads>0</read_threads>                       <!-- Number of I/O Read Threads -->
   <write_threads>0</write_threads>                     <!-- Number of I/O Write Threads -->
   <scan_threads>0</scan_threads>                       <!-- Number of Dir Scanning Threads
-->
   <meta_threads>0</meta_threads>                       <!-- Number of Metadata Threads -->
   <worker_threads>0</worker_threads>
   <sparse_file>false</sparse_file>                     <!-- Sparse File Checking -->
   <fail_on_attr_error>yes</fail_on_attr_error>         <!-- Behavior on Attr Error -->
   <compression_method>lz4</compression_method>         <!-- Compression Method for File
Transfer -->
   <use_file_cache>true</use_file_cache>               <!-- Use File Cache -->
   <max_file_cache_buffer>0</max_file_cache_buffer>    <!-- Max File Cache Buffer-->
   <resume_suffix>.aspx</resume_suffix>                <!-- Resume Suffix -->
   <symbolic_links>follow,create</symbolic_links>      <!-- Symbolic Link Actions -->
   <preserve_attributes> </preserve_attributes>        <!-- Preserve Attributes -->
   <overwrite>allow</overwrite>                        <!-- Overwrite -->
   <file_manifest>disable</file_manifest>              <!-- File Manifest -->
   <file_manifest_path>path</file_manifest_path>        <!-- File Manifest Path -->
   <file_manifest_inprogress_suffix>.aspera-inprogress</file_manifest_inprogress_suffix>
                                                       <!-- File Manifest Suffix -->
   <pre_calculate_job_size>any</pre_calculate_job_size><!-- Pre-Calculate Job Size -->
   <replace_illegal_chars></replace_illegal_chars>     <!-- Convert Restricted Windows
Characters -->
   <storage_rc>
      <adaptive>true</adaptive>                        <!-- Storage Rate Control -->
   </storage_rc>
   <filters>                                           <!-- File Filter Pattern List -->
      <filter>rule1</filter>
      <filter>rule2</filter>
   </filters>
   <file_create_mode> </file_create_mode>              <!-- File Create Mode -->
   <file_create_grant_mask>644</file_create_grant_mask><!-- File Create Grant Mask -->
   <directory_create_mode> </directory_create_mode>    <!-- Directory Create Mode -->
```

```
    <directory_create_grant_mask>755</directory_create_grant_mask>
                                                    <!-- Directory Create Grant Mask -->
    <partial_file_suffix>.partial</partial_file_suffix> <!-- Partial File Suffix -->
    <file_checksum>any</file_checksum>                  <!-- File Checksum Method -->
</file_system>
```

3. Edit settings as needed.

File System Settings Reference

Field	Description	Values	Default
Absolute Path	The absolute path, or docroot, is the area of the file system that is accessible to an Aspera transfer user. The default empty value allows access to the entire file system. You can set one global docroot and then further restrict access to the file system by group or individual user. Docroot paths require specific formatting depending on where the transfer server's storage is located. **Format examples** • Local storage absolute path: `/home/aspera424/movies` Or using a placeholder for usernames: `/home/$(name)` • Local storage in URI format: `file:///home/bear/movies` URI format is required for server-side encryption-at-rest, but is not supported by the Aspera Watch Service. Aspera recommends setting a global docroot to an empty folder or a part of the file system specific to each user. If there is a pattern in the docroot of each user, for example, *username*, you can use a substitutional string. This allows you to assign an independent docroot to each user without setting it individually for each user. See “Setting Up Users” on page 20 for information. You can also set multiple docroots and make them conditional based on the IP address from which the connection is made by editing `aspera.conf`. To do so, edit the absolute path setting by adding the IP address using the following syntax: `<absolute peer_ip="ip_address">path</absolute>` Growing files support allows you to start transferring files to the target directory while they are still being written to the source directory. To configure `aspera.conf` for growing files: • Edit the `<absolute>` section.	file path or URI	undefined (total access)

Field	Description	Values	Default
	• Add your growing files specification using the syntax described in “Ascp Command Reference” on page 121 for the *source* element. See also “Ascp General Examples” on page 136.		

File Restriction	**Note:** A configuration (global, group, or user) can have a docroot or a file restriction; configurations with both are not supported.
	A set of file system filters that use "*" as a wildcard and "!" to indicate "exclude". Paths are in URI format; special characters in a URI must be URL-encoded.
	Access to a file is rejected unless the file matches the restrictions, which are processed in the following order:
	• If a restriction starts with "!", the user is not allowed to access any files that match the rest of the restriction.
	• If a restriction does not start with "!", the user can access any file that matches the filter.
	• If one or more restrictions do not start with "!", the user can access any file that matches any one of the no-"!" restrictions.
	Format examples:
	• For a specific folder:
	`file:////docs/*`
	• For the drive root:
	`file:///c*`
	• For ICOS-S3 storage:
	`s3://my_vault/*`
	• To exclude access to key files:
	`!*.key`
		URI	undefined (total access)

Read Allowed	Set to `true` (default) to allow users to transfer files and folders from their docroot.	• `true` • `false`	`true`
Write Allowed	Set to `true` (default) to allow users to transfer files and folders to their docroot.	• `true` • `false`	`true`
Browse Allowed	Set to `true` (default) to allow users to browse their docroot.	• `true` • `false`	`true`
Read Block Size (bytes)	Set the maximum number of bytes that can be stored within a block as the block is being transferred from the source disk drive to the receiver. The default of zero causes the Aspera sender to use its default internal buffer size,	positive integer, where 500MB or 524,288,0	0

Field	Description	Values	Default
	which may vary by operating system. This is a performance-tuning parameter for an Aspera sender (which only takes effect if the *sender* is a server).	00 bytes is the maximum block size.	
Write Block Size (bytes)	Set the maximum bytes within a block that an **ascp** receiver can write to disk. The default of zero causes the Aspera receiver to use its default internal buffer size, which may vary by operating system. This is a performance-tuning parameter for an Aspera receiver (which only takes effect if the *receiver* is a server).	positive integer, where 500MB or 524,288,000 bytes is the maximum block size.	0
Number of I/O read threads	Set the number of threads the Aspera sender uses to read file contents from the source disk drive. It takes effect on both client and server, when acting as a sender. The default of zero causes the Aspera sender to use its internal default, which may vary by operating system. This is a performance-tuning parameter for an Aspera sender.	positive integer	0
Number of I/O Write Threads	Set the number of threads the Aspera receiver uses to write the file contents to the destination disk drive. It takes effect on both client and server, when acting as a receiver. The default of zero causes the Aspera receiver to use its internal default, which may vary by operating system. This is a performance-tuning parameter for an Aspera receiver.	positive integer	0
Number of Dir Scanning Threads	Set the number of threads the Aspera sender uses to scan directory contents. It takes effect on both client and server, when acting as a sender. The default of zero causes the Aspera sender to use its internal default. This is a performance-tuning parameter for an Aspera sender.	positive integer	0
Number of Metadata Threads	Set the number of threads the Aspera receiver uses to create directories or 0 byte files. It takes effect on both client and server, when acting as a receiver. The default of zero causes the Aspera receiver to use its internal default, which may vary by operating system. This is a performance-tuning parameter for an Aspera receiver.	positive integer	0
Number of Worker Threads	Set the number of threads the Aspera sender and receiver use to delete files. This is a performance-tuning parameter.	positive integer	0
Sparse File Checking	Set to `true` to enable sparse file checking, which tells the Aspera receiver to avoid writing zero blocks and save disk space. The default of `false` to tell the Aspera receiver to write all the blocks. This is a performance-tuning parameter for an Aspera receiver.	`true` or `false`	`false`

Field	Description	Values	Default
Behavior on Attr Error	Set behavior for when operations attempt to set or change file attributes (such as POSIX ownership, ACLs, or modification time) and fail. Setting to `yes` returns an error and causes the operation to fail. Setting to `no` logs the error and the operation continues	`no` or `yes`	`yes`
Compression Method for File Transfer	Set the compression method to apply to transfers. It applies to both the client and server.	`lz4`, `qlz`, `zlib`, or `none`	`lz4`
Use File Cache	Set to `true` (default) to enable per-file memory caching at the data receiver. File level memory caching improves data write speed on Windows platforms in particular, but uses more memory. This is a performance tuning parameter for an Aspera receiver. Aspera suggests using a file cache on systems that are transferring data at speeds close to the performance of their storage device, and disable it for system with very high concurrency (because memory utilization will grow with the number of concurrent transfers).	`true` or `false`	`true`
Max File Cache Buffer (bytes)	Set the maximum size allocated for per-file memory cache (see Use File Cache) in bytes. The default of zero will cause the Aspera receiver to use its internal buffer size, which may be different for different operating systems. This is a performance tuning parameter for an Aspera receiver.	positive integer	`0`
Resume Suffix	Set the file name extension for temporary metadata files used for resuming incomplete transfers. Each data file in progress will have a corresponding metadata file with the same name plus the resume suffix specified by the receiver. Metadata files in the source of a directory transfer are skipped if they end with the sender's resume suffix.	text string	`.aspx`
Symbolic Link Actions	Set how the server handles symbolic links. For more information about the actions and the interaction between the server configuration and the client request, see “Symbolic Link Handling” on page 150. Combinations of values are logically ORed before use. For example, use `none` alone to mean skip, and shut out other options; when both `follow` and `follow_wide` are set, the latter is recognized.	`none`, `create`, `follow`, `follow_wide`, or any combination of the above delimited by commas	`follow,create`
Preserve Attributes	Set the file creation policy. Set to `none` to not preserve the timestamps of source files. Set to	`none` or `times`	blank (use the client setting)

Field	Description	Values	Default
	`times` to preserve the timestamp of the source files at destination. **Note:** For Limelight storage, only the preservation of modification time is supported.		
Overwrite	Set to `allow` to allow Aspera clients to overwrite existing files on the server, as long as file permissions allow that action. If set to `deny`, clients who upload files to the server cannot overwrite existing files, regardless of file permissions.	`allow` or `deny`	`allow`
File Manifest	Set to `text` to generate a text file "receipt" of all files within each transfer session. Set to `disable` to not create a File Manifest. The file manifest is a file containing a list of everything that was transferred in a given transfer session. The filename of the File Manifest itself is automatically generated based on the transfer session's unique ID. The location where each manifest is written is specified by the File Manifest Path value. If no File Manifest Path is specified, the file will be generated under the destination path at the receiver, and under the first source path at the sender.	`text`, `disable`, or `none`	`none`
File Manifest Path	Specify the location to store manifest files. Can be an absolute path or a path relative to the transfer user's home. **Note:** File manifests can only be stored locally. Thus, if you are using S3, or other non-local storage, you must specify a *local* manifest path.	text string	blank
File Manifest Suffix	Specify the suffix of the manifest file during file transfer.	text string	`.aspera-inprogress`
Pre-Calculate Job Size	Set to `yes` to enable calculating job size before transferring. Set to `no` to disable calculating job size before transferring. Set to `any` to follow client configurations.	`yes`, `no`, or `any`	`any`
Convert Restricted Windows Characters	To enable the replacement of reserved Windows characters in file and directory names with a non-reserved character, set to the single byte, non-restricted character that will be used for the replacement. Only applies to files written to the local Windows file system; to enable on the peer it must be set on the peer's system.	single-byte, non-restricted character	blank
File Create Mode	Set the file creation mode (permissions). If specified, create files with these permissions (for example, 0755), irrespective of File Create Grant Mask and permissions of the file on the source computer. Only takes effect when the server is a non-Windows receiver.	positive integer (octal)	undefined

Field	Description	Values	Default
File Create Grant Mask	Set the mode for newly created files if File Create Mode is not specified. If specified, file modes will be set to their original modes plus the Grant Mask values. Only takes effect when the server is a non-Windows receiver and when File Create Mode is not specified.	positive integer (octal)	644
Directory Create Mode	Set the directory creation mode (permissions). If specified, create directories with these permissions irrespective of Directory Create Grant Mask and permissions of the directory on the source computer. Only takes effect when the server is a non-Windows receiver.	positive integer (octal)	undefined
Directory Create Grant Mask	Set the mode for newly created directories if Directory Create Mode is not specified. If specified, directory modes will be set to their original modes plus the Grant Mask values. Only takes effect when the server is a non-Windows receiver and when Directory Create Mode is not specified.	positive integer (octal)	755
File Filter Pattern List	Exclude or include files and directories with the specified pattern in the transfer. Add multiple entries for more inclusion/exclusion patterns. To specify an inclusion, start the pattern with '+ ' (+ and a whitespace). To specify an exclusion, start the pattern with '- ' (- and a whitespace). Two symbols can be used in the setting of patterns: • A "*****" (asterisk) represents zero to many characters in a string. For example, `*.tmp` matches `.tmp` and `abcde.tmp`. • A "**?**" (question mark) represents a single character. For example, `t?p` matches `tmp` but not `temp`. For details on specifying rules, see "Using Filters to Include and Exclude Files" on page 145. This option applies only when the server is acting as a client. Servers cannot exclude files or directories uploaded or downloaded by remote clients.	text entries	blank
Partial File Name Suffix	Set the filename extension on the destination computer while the file is being transferred. Once the file has been completely transferred, this filename extension is removed. **Note:** This option only takes effect when it is set on the receiver side.	text string	blank
File Checksum Method	Set the type of checksum to calculate for transferred files. The content of transfers can be verified by comparing the checksum value at the destination with the value read at the source. For more information, see "Reporting Checksums" on page 55.	any, `md5`, `sha1`, `sha256`, `sha384`, or `sha512`	any

4. Save and validate `aspera.conf`.

   Run the following command to confirm that the XML is correctly formatted and the parameter settings are valid:

```
# /opt/aspera/bin/asuserdata -v
```

aspera.conf - Transfer Server Configuration

The settings in the <central_server> section of `aspera.conf` include the network and port that asperacentral uses to process transfer requests and how to manage the asperacentral database.

About this task

Configuration methods: These instructions describe how to manually modify `aspera.conf`. You can also add and edit these parameters using **asconfigurator** commands. For more information on using **asconfigurator**, see "User, Group and Default Configurations" on page 328 and run the following command to retrieve a complete default `aspera.conf` that includes the **asconfigurator** syntax for each setting:

```
# /opt/aspera/bin/asuserdata -+
```

Procedure

1. Open `aspera.conf` from the following location:

   `/opt/aspera/etc/aspera.conf`
2. Add or locate the <central_server/> section, as shown in the following example:

```
<central_server>
  <address>127.0.0.1</address>                                    <!-- Address -->
  <port>40001</port>                                              <!-- Port -->
  <persistent_store>enable</persistent_store>                     <!-- Persistent Storage -->
  <files_per_session>1000</files_per_session>                     <!-- Files Per Session -->
  <persistent_store_path></persistent_store_path>                 <!-- Persistent Storage Path
-->
  <persistent_store_max_age>86400</persistent_store_max_age>      <!-- Maximum Age -->
  <persistent_store_on_error>ignore</persistent_store_on_error> <!-- Exit Central on Storage
Error -->
  <compact_on_startup>enable</compact_on_startup>                 <!-- Compact Database on
Startup -->
  <ignore_empty_files>true</ignore_empty_files>                   <!-- Ignore Empty Files -->
  <ignore_no_transfer_files>true</ignore_no_transfer_files>       <!-- Ignore No-transfer
Files -->
  <validation_timeout>300</validation_timeout>                    <!-- Post-Transfer
Validation Timeout -->
</central_server>
```

3. Edit settings as needed.

   Central Server Settings Reference

Setting	Description	Values	Default
Address	The network interface address on which the transfer server listens. The default value of 127.0.0.1 enables the transfer server to accept transfer requests from the local computer. If you set the address to 0.0.0.0, the transfer server can accept requests on all network interfaces. Alternatively, a specific network interface address may be specified.	Valid IPv4 address	127.0.0.1
Port	The port on which the transfer server accepts transfer requests.	Positive integer 1 - 65535	40001

Setting	Description	Values	Default
Persistent Storage	Enable to retain data that is stored in the database between reboots of asperacentral.	Enable or Disable	Enable
Files Per Session	The maximum number of files that can be retained for persistent storage.	Positive integer	1000
Persistent Storage Path	The location in which to store data between reboots of asperacentral. If the path is a directory, then a file is created with the default name `central-store.db`. Otherwise, the file is named as specified in the path.	Valid system path	If the application is installed in the default location, then the path is the following: `/opt/aspera/var/`
Maximum Age (seconds)	Maximum allowable age (in seconds) of data to be retained in the database.	Positive integer	86400
Exit Central on Storage Error	The behavior of the asperacentral server if a database write error occurs.	Ignore or Exit	Ignore
Compact Database on Startup	Enable or disable compacting (vacuuming) the database when the transfer server starts.	Enable or Disable	Enable
Ignore Empty Files	Set to **true** to block the logging of zero-byte files.	true or false	true
Ignore No-transfer Files	Set to **true** to block the logging of files that were not transferred because they exist at the destination.	true or false	true
Post-Transfer Validation Timeout	How many seconds to wait for a post-transfer validator to update the status of a file before the file is released from the validator and its status is changed back to "to_be_validated". This allows a file to be validated by a different validator if the first validator stops working. For more information, see “Out-of-Transfer File Validation” on page 109.	Positive integer	300

4. Save and validate `aspera.conf`.

   Run the following command to confirm that the XML is correctly formatted and the parameter settings are valid:

```
# /opt/aspera/bin/asuserdata -v
```

aspera.conf - 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.

Filters can be specified on the **ascp** command line and in `aspera.conf`. Ascp applies filtering rules that are set in `aspera.conf` *before* it applies rules on the command line.

The **ascp -N** and **-E** options let you specify filter rules individually for each transfer, while filter options configured in `aspera.conf` allow you to have the same rules applied to all transfers.

Filter rules that **ascp** finds in `aspera.conf` are always applied before any command-line rules. This allows you to specify individual command-line rules to augment a core set specified in `aspera.conf`.

Rule Syntax

A rule consists of a "+" or "-" sign (indicating whether to include or exclude), followed by a space character, followed by a pattern. A pattern can be a file or directory name, or a set of names expressed with UNIX *glob* patterns.

Basic usage

- Filtering rules are applied to the transfer list in the order that they are listed in `aspera.conf`.
- Filtering is a process of exclusion, and include rules override exclude rules that follow them. Include rules cannot add back files that are excluded by a preceding exclude rule.
- Include rules must be followed by at least one exclude rule, otherwise all files are transferred because none are excluded. To exclude all unmatched files, add two final rules: "`- *`" and "`- .*`".
- Filtering operates only on the set of files and directories in the transfer list. An include rule cannot add files or directories that are not already part of the transfer list.

Example	Transfer Result
`-` *rule*	Transfer all files and directories except those with names that match *rule*.
`+` *rule*	Transfer all files and directories because none are excluded.
`+` *rule1* `-` *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*.
`-` *rule1* `+` *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*`'`. Additional filters can be set for or on the command line (“Using Filters to Include and Exclude Files” on page 145).

Filtering Rule Application

Filtering order

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

1. The first file (or directory) in the transfer list is compared to the pattern of the first rule.
2. If the file matches the pattern, Ascp includes it or excludes it 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.
3. If the file does not match, it is compared to 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.

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`.

For details on using wildcards and special characters to build rule patterns, see "Using Filters to Include and Exclude Files" on page 145.

Set Rules

Filter rules can be set in `aspera.conf` in the following ways:

- by modifying `aspera.conf` with the **`asconfigurator`** tool
- by modifying `aspera.conf` directly with a text editor

In order to run **`asconfigurator`** successfully, you must meet the following requirements:

1. have write access to `aspera.conf`
2. not be restricted to **`aspshell`**, which does not allow running **`asconfigurator`**

The set commands for user, group, and global filter settings use the following syntax:

```
asconfigurator -x "set_user_data;user_name,username;file_filters,|rule1|rule2...|ruleN"
asconfigurator -x "set_group_data;group_name,groupname;file_filters,|rule1|rule2...|ruleN"
asconfigurator -x "set_node_data;file_filters,|rule1|rule2...|ruleN"
```

Where:

- Each rule argument, including the first, must begin with a "|" character, which serves as the separator between multiple rules.
- To clear rules, run **`asconfigurator`** by specifying "`file_filters,`" without rule arguments. Note that the comma in "`file_filters,`" is still required. See the example below.
- Running **`asconfigurator`** replaces the specified settings; it does not add to them.

To edit `aspera.conf`, open it from the following location:

`/opt/aspera/etc/aspera.conf`

See the following examples for the correct syntax.

Examples

- Set global include and exclude filters:

```
# asconfigurator -x "set_node_data;file_filters,|+ file.txt|- *.txt"
```

  Results in `aspera.conf`:

```
    <default>
        <file_system>
            <filters>
                <filter>+ file.txt</filter>
          <filter>- *.txt</filter>
            </filters>
        </file_system>
    </default>
```

- Sets filters for user `asp1`:

```
# asconfigurator -x "set_user_data;user_name,asp1;file_filters,|+ abc/wxy/tuv/**|- abc/**/def"
```

  Results in `aspera.conf`:

```
    <aaa>
        <realms>
            <realm>
                <users>
```

```
                    <user>
                        <name>asp1</name>
                        <file_system>
                            <filters>
                                <filter>+ abc/wxy/tuv/**</filter>
                                <filter>- abc/**/def</filter>
                            </filters>
                        </file_system>
                    </user>
                </users>
            </realm>
        </realms>
    </aaa>
```

- Clears all filters for the group `asgroup`:

```
# asconfigurator -x "set_group_data;group_name,asgroup;file_filters,"
```

Results in `aspera.conf`:

```
    <groups>
        <group>
            <name>asgroup</name>
            <file_system>
                <filters />
            </file_system>
        </group>
    </groups>
```

Server-Side Encryption-at-Rest (EAR)

When files are uploaded from an Aspera client to HSTS, server-side encryption-at-rest (EAR) saves files on disk in an encrypted state. When downloaded from HSTS, server-side EAR first decrypts files automatically, and then the transferred files are written to the client's disk in an unencrypted state.

Capabilities

Server-side EAR provides the following advantages:

- It protects files against attackers who might gain access to server-side storage. This is important primarily when using NAS storage or cloud storage, where the storage can be accessed directly (and not just through the computer running HSTS or HSTE).
- It is especially suited for cases where the server is used as a temporary location–for example, when a client uploads a file and another one downloads it.
- Server-side EAR can be used together with client-side EAR. When used together, content is doubly encrypted. For more information, see “Client-Side Encryption-at-Rest (EAR)” on page 156.
- Server-side EAR doesn't create an "envelope" as client-side EAR does. The transferred file stays the same size as the original file. The server stores the metadata necessary for server-side EAR separately in a file of the same name with the file extension `.aspera-meta`. By contrast, client-side EAR creates a envelope file containing both the encrypted contents of the file and the encryption metadata, and it also changes the name of the file by adding the file extension `.aspera-env`.
- It works with both regular transfers (FASP) and HTTP fallback transfers.

Requirements

If the following requirements are not met, then the server can have both encrypted and unencrypted content. This can cause file corruption on the server or unintended overwriting of downloaded files on the client.

- Server-side EAR must be configured when the server is first set up.
- When multiple users have access to the same area of the file system, they must use the same EAR configuration.

Limitations and Considerations

- Server-side EAR is not designed for cases where files need to move in an encrypted state between multiple computers. For that purpose, client-side EAR is more suitable: files are encrypted when they first leave the client, then stay encrypted as they move between other computers, and are decrypted when they reach the final destination and the passphrase is available.
- Server-side EAR does not work with multi-session transfers (using **ascp -C** or Node API `multi_session` set to greater than 1).
- Do not mix server-side EAR and non-EAR files in transfers, which can happen if server-side EAR is enabled after the server is in use or if multiple users have access to the same area of the file system but have different EAR configurations.

Configuring Server-Side EAR

Procedure

1. Set the docroot in URI format.

   Server-side EAR requires the storage to have a docroot in URI format, such that it is prefixed with `file:///`. The third slash (/) does not serve as the root slash for an absolute path. For example, a docroot of `/home/xfer` would be set as `file:////home/xfer` and a docroot of `C%3A\Users\xfer` would be set as `file:///C%3A\Users\xfer`.

   To set the docroot for a user, group, or default from the command line, run the appropriate **asconfigurator** command:

   ```
   # asconfigurator -x "set_user_data;user_name,username;absolute,file:///filepath"
   # asconfigurator -x "set_group_data;group_name,group_name;absolute,file:///filepath"
   # asconfigurator -x "set_node_data;absolute,file:///filepath"
   ```

2. Set the password.

   The server-side EAR password can be set for all users (global), per group, or per user. Set the password by using **asconfigurator** or manually editing `aspera.conf`:

   To set the EAR password for a user, group, or default, run the appropriate command:

   ```
   # asconfigurator -x
   "set_user_data;user_name,username;transfer_encryption_content_protection_secret,passphrase"
   # asconfigurator -x
   "set_group_data;group_name,group_name;transfer_encryption_content_protection_secret,passphras
   e"
   # asconfigurator -x "set_node_data;transfer_encryption_content_protection_secret,passphrase"
   ```

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 HSTS or HSTE 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
```

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 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.

Server Logging Configuration for Ascp and Ascp4

Server transfer logs are stored in the default location (see “Log Files” on page 356), rotated once they are 10 MB, and log at "log" level. For **ascp** transfers, you can configure a different default log directory, log size, number of logs, and logging intensity on the server, and apply these settings globally or to specific users. For Ascp4 transfers, you can configure a default log size (Ascp4 does not support user-specific logging settings).

About this task

If the client specifies a log directory on the server (using `-R` *`remote_log_dir`*) or the location and size of the local log directory (using `-L` *`local_log_dir`*`[:`*`size`*`]`), then these take precedence over the server settings.

Default vs User-specific Settings

You can set the default logging configuration or assign users to different logging classes, which are sets of logging configurations.

Note: Default settings override user-specific settings. To enable user-specific settings, do not set default settings. User settings do not apply to Ascp4 transfers.

Configuration Methods

Logging settings are configured by running **asconfigurator** commands (recommended) or by manually editing `aspera.conf`. To edit `aspera.conf`, open it with admin privileges from the following location:

`/opt/aspera/etc/aspera.conf`

Procedure

1. To set default logging values, run the following commands, as required:

```
# asconfigurator -x "set_logging_data;directory,logging_directory"
# asconfigurator -x "set_logging_data;log_size,size_mb"
# asconfigurator -x "set_logging_data;log_count,count"
# asconfigurator -x "set_logging_data;level,log_level"
```

Object	Description
`directory`	The full path to the logging directory. Applies only to **ascp** transfers.
`log_size`	The size of the log file, in MB, at which it is rotated (the oldest information is overwritten by the newest information). Default: 10 MB. Applies to **ascp** and **ascp4** transfers.

Object	Description
log_count	The number of log files to be retained. The default is 10, and the maximum is 250. If a value of zero or a negative number is provided, it is ignored (and the default of 10 is used).
level	The logging level. Valid values are log (default), dbg1, or dbg2. Applies only to **ascp** transfers.

These commands modify the <logging> sub-section of the <default> section of aspera.conf (or you can manually edit the file):

```
...
<default>
    <file_system>...</file_system>
    <logging>
        <directory>logging_directory</directory>
        <log_size>size_mb</log_size>
        <log_count>count</log_count>
        <level>log_level</level>
    </logging>
</default>
...
```

2. To set user logging values, create logging classes (each with a specific logging configuration) and then assign users to classes.

   a) Create a logging class:

```
# asconfigurator -x
"set_log_setting_data;classes,class_name;directory,logging_directory;log_size,size_mb;leve
l,log_level"
```

Object	Description
classes	The name of the class. This is the value that you use to assign users to this "class" of logging settings.
directory	The full path to the logging directory. Applies only to **ascp** transfers.
log_size	The size of the log file, in MB, at which it is rotated (the oldest information is overwritten by the newest information). Default: 10 MB. Applies to **ascp** and **ascp4** transfers.
level	The logging level. Valid values are log (default), dbg1, or dbg2. Applies only to **ascp** transfers.

   b) Assign a user to the logging class:

```
# asconfigurator -x "set_user_data;user_name,username;logging_class,class_name"
```

For example, the following commands create two logging classes, admin and home. The home logging class uses the substitution string $(home) to log to the user's home directory, ensuring that the transfer users have access to the log files for their transfers. They assign user root to the admin logging configuration, and users user1 and user2 to the home logging configuration.

```
# asconfigurator -x "set_log_setting_data;classes,admin;directory,/root/
logs;log_size,3;level,dbg"
# asconfigurator -x "set_log_setting_data;classes,home;directory,$(home)/
logs;log_size,20";level,dbg"
# asconfigurator -x "set_user_data;user_name,root;logging_class,admin"
# asconfigurator -x "set_user_data;user_name,user1;logging_class,home"
```

This created the following in aspera.conf:

```
...
<logging>
    <log_setting>
```

```
        <classes>admin</classes>
        <directory>/root/logs</directory>
        <log_size>3</log_size>
        <level>dbg</level>
    </log_setting>
    <log_setting>
        <classes>home</classes>
        <directory>$(home)/logs</directory>
        <log_size>20</log_size>
        <level>log</level>
    </log_setting>
</logging>
<aaa><realms><realm>
    <users>
        <user>
           <name>root</name>
           <logging_class>admin</logging_class>
           <file_system>...</file_system>
        </user>
        <user>
           <name>user1</name>
           <logging_class>home</logging_class>
           <file_system>...</file_system>
        </user>
        <user>
           <name>user2</name>
           <logging_class>home</logging_class>
           <file_system>...</file_system>
        </user>
    </users></realm></realms>
</aaa>
...
```

3. If you manually edited `aspera.conf`, save your changes.
4. If you manually edited `aspera.conf`, validate the XML form of `aspera.conf`:

```
# /opt/aspera/bin/asuserdata -v
```

Out-of-Transfer File Validation

Out-of-transfer file validation is run as soon as the client uploads a file to HSTS. The transfer is reported as complete and then the validation is run. The validation script uses the Aspera Reliable Query API to retrieve the list of files to validate and update the file status during validation. The transfer user who is transferring files to the server must be associated with Node API user credentials in order to use the API. These instructions describe how to associate a transfer user with Node API user credentials, create a validation script, and configure the server to use out-of-transfer file validation on files that it receives from specific transfer users, groups, or globally.

About this task

This approach has several benefits over inline file validation:

- More efficient use of system resources because the **ascp** sessions can close before validation is completed.
- Out-of-transfer file validation is applied to transfers that use HTTP(S) fallback transport.
- Files are explicitly reported as "validating" to IBM Aspera Faspex through asperacentral. Files that are validated inline are reported as "transferring".

Procedure

1. Associate the transfer user with a Node API username and password, if not already configured.

```
# /opt/aspera/bin/asnodeadmin -a -u node_username -p node_password -x transfer_user
```

To view existing Node API users and the transfer users associated with them, run the following command:

```
# /opt/aspera/bin/asnodeadmin -l
```

2. Create your validation script.

   Note: The validation service must be executed on a system that has access to the storage.

   The validation script should follow these steps:

   a) Identify the files that need to be validated by using the Reliable Query REST API:

   ```
   curl -X POST -u node_user:password -d '{ "file_transfer_filter": { "max_result": 20},
   "validation": { "validator_id": "validator_id" } }' https://server_name:9092/services/
   rest/transfers/v1/files
   ```

   Where the *validator_id* is a unique ID to prevent simultaneous validation of the same file by different validators. The value for `max_result` sets a "batch size" for how many files are collected for validation by each POST request, and cannot exceed 1000.

   The POST request retrieves the files that are "to_be_validated", updates their state to "validating" and the owner to the validator ID, and returns the file list, with information similar to the following:

   ```
   {
      "file_transfer_info_result": {
           "file_transfer_info" : [
               { "session_uuid":"9a2678c3-64db-4bc1-abd4-605ad7702230",
                 "path" :"/tmp/src/dir","local_id":1,
                 "file_id":"47203042-bb57-487f-95df-ad614d0a3720",
                  "status":"validating",
                  "new_file":true, "error_code":0,
                  "size":10000000,
                  "start_offset":0,
                  "bytes_written":10000000,
                  "bytes_contiguous":0, "bytes_lost":0, "elapsed":0,"bytes_processed":0,
                  "start_date":"2017-11-29T16:21:24Z",
                  "checksum_type":"None"
    } ],
   "iteration_token":"0000000000000003",
   "remaining_result_count":1,
   "result_count":1
   } }
   ```

   b) Validate the files and update the "bytes_processed".

   By updating the "bytes_processed", the GUI can display a progress bar:

   ```
   curl -X PUT -u node_user:password -d '{ "validator_id": "validator_id", "files":
   [{ "session_uuid": "session_uuid", "file_id": "file_id", "status": "validating",
   "bytes_processed": bytes } ] }' https://server_name:9092/services/rest/transfers/v1/files
   ```

   Note: If a validator does not update the file status within the validation timeout, the file status is reset to "to_be_validated" and the file is released from the validator so that the file can be validated by a different validator. The default timeout is 5 minutes. To edit the validation timeout, run the following command:

   ```
   # asconfigurator -x "set_central_server_data;validation_timeout,seconds"
   ```

   c) Update the status of each file as validation completes or fails:

   If a file passes validation, update its status to "completed":

   ```
   curl -X PUT -u node_user:password -d '{ "validator_id": "validator_id", "files":
   [{ "session_uuid": "session_uuid", "file_id": "file_id", "status": "completed" } ] }'
   https://server_name:9092/services/rest/transfers/v1/files
   ```

   If the file fails validation, update its status to "error" and provide an error code (as a number) and error description (as a string):

   ```
   curl -X PUT -u node_user:password -d '{ "validator_id": "validator_id", "files":
   [{ "session_uuid": "session_uuid", "file_id": "file_id", "status": "error", "error_code":
   ```

```
error_number, "error_description": "error_string" } ] }' https://server_name:9092/
services/rest/transfers/v1/files
```

For example, the body of a PUT request could contain the following information for three files:

```
{
    "validator_id": "my identifier",
    "files": [
       {
    "session_uuid": "1425c741-32bb-492d-b5e1-724c8bdb1fbf",
    "file_id": "11111111-11422dfb-5b8ed464-239783b8-09c78597",
    "status": "validating",
    "bytes_processed": 3
       },
       {
    "session_uuid": "1425c741-32bb-492d-b5e1-724c8bdb1fbf",
    "file_id": "22222222-11422dfb-5b8ed464-239783b8-09c78597",
    "status": "completed"
       },
       {
    "session_uuid": "1425c741-32bb-492d-b5e1-724c8bdb1fbf",
    "file_id": "33333333-11422dfb-5b8ed464-239783b8-09c78597",
    "status": "error",
          "error_code": 2,
          "error_description": "File not found"
       }
    ]
}
```

If all files validate and update successfully, HTTP 204 is returned. If one or more files have failed validation, HTTP 200 is returned. For each failed file, an entry is added to the result. If another HTTP code is returned, then a more general error, such as invalid JSON, has occurred.

3. Confirm that persistent storage is enabled (the default setting).

   From the command line, run the following command:

   ```
   # /opt/aspera/bin/asuserdata -c
   ```

   In the output, locate the value for "persistent_store". If it is not set to "enable", run the following command:

   ```
   # asconfigurator -x "set_central_server_data;persistent_store,enable"
   ```

4. Ensure that empty files and files that exist at the destination (and are skipped by the transfer session) are not ignored.

   From the command line, run the following command:

   ```
   # asconfigurator -x "set_central_server_data;ignore_no_transfer_files,false"
   ```

   If ignore_no_transfer_files is set to true, the workflow might fail when the transfer attempts to create empty files on the destination and they are not validated.

5. Schedule the validation.

   The validation can be scheduled for one or more users (files that are transferred to the server by those users are validated), for one or more groups (files that are transferred to the server by users in the groups are validated), or globally (all files that are transferred to the server for all users are validated).

   From the command line, run the command corresponding to the scope of your configuration:

   ```
   # asconfigurator -x "set_user_data;user_name,username;validation_file_stop,post_transfer"
   # asconfigurator -x "set_group_data;group_name,groupname;validation_file_stop,post_transfer"
   # asconfigurator -x "set_node_data;validation_file_stop,post_transfer"
   ```

Inline File Validation

If an executable file containing malicious code is uploaded to the server, the malicious code can subsequently be executed by an external product that integrates with an Aspera product. Inline file validation is a feature that enables file content to be validated while the file is in transit, as well as when the transfer is complete. The validation check can be made with a Lua script, or with a REST call to an

external URL. The mode of validation used (URL or Lua) and the timing of the check are set in `aspera.conf`.

About this task

When URI inline file validation is enabled, the transfer is not reported as complete until the validation completes. An alternative to inline file validation, out-of-transfer file validation, completes the transfer and then validates the file, and can be substantially faster. For more information, see "Out-of-Transfer File Validation" on page 109.

Lua scripting is supported for many uses, including inline file validation. For detailed information, see "Automated Execution of Lua Scripts with Transfer Events" on page 177.

Note: Inline file validation is not applied to transfers that fall back to HTTP. If all transfers require validation, use out-of-transfer validation.

Procedure

1. For URI validation, configure the REST service and set the URL.

   Note: The code examples provided here are for an admin using a Java servlet deployed on an Apache web server, but this process is generalizable to other programming languages and other servers.

   a) Open `web.xml` and edit the `<servlet>` and `<servlet_mapping>` sections to provide the necessary information for validation.

   The `<servlet-name>` (URL handler) value is also configured in `aspera.conf` (in the next step) and any custom code (such as file filtering, see "Inline File Validation with URI" on page 113).

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://
   xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
            version="3.1">

       <servlet>
           <servlet-name>SimpleValidator</servlet-name>
           <servlet-class>aspera.validation.SimpleValidator</servlet-class>
       </servlet>

       <servlet-mapping>
           <servlet-name>SimpleValidator</servlet-name>
           <url-pattern>/SimpleValidator/validation/files</url-pattern>
       </servlet-mapping>
   </web-app>
   ```

   b) Set the URL in `aspera.conf`.

   ```
   # asconfigurator -x "set_user_data;user_name,username;validation_uri,url"
   ```

   Where *url* is the server's IP address and port, and the servlet name (URL handler) found in `web.xml`. This adds the path to the `<transfer>` section of `aspera.conf`. For example:

   ```
   <transfer>
   <validation_uri>http://127.0.0.1:8080/SimpleValidator</validation_uri>
   </transfer>
   ```

2. Schedule the validation.

   To define URI validation from the command line, run this command:

   ```
   # asconfigurator -x "set_user_data;user_name,username;validation_threshold,{uri}"
   ```

   (You can set a Lua script validation to run at one event and a URI validation to run at another, but you can define only one Lua script or URL. The default setting for all events is none.)

3. If you schedule validation at a file size threshold, set the threshold.

```
# asconfigurator -x "set_user_data;user_name,username;validation_threshold_kb,size"
```

4. Configure multi-threaded validation.

   By default, inline validation is set to use 5 threads.

   If the number of validation threads is not set to 1, then multiple threads may perform different types of validations for different (or the same) files at the same time. In such a situation, the response of a `validation_file_stop` at the end of a file download might come before the response of a `validation_threshold` for the same file.

   To set the number of validation threads, run the following command:

```
# asconfigurator -x "set_user_data;user_name,username;validation_threads,number"
```

Results

For more information about the configuration parameters, see “aspera.conf - Transfer Configuration” on page 70 (defining values in `aspera.conf`)

For more information on the output of your inline validation, see “Inline File Validation with URI” on page 113 or “Inline File Validation with Lua Script” on page 115.

Inline File Validation with URI

Inline file validation with URI can be customized to filter which files are validated.

Validation Requests and Returned Responses

During the inline validation process, **ascp** automatically generates a JSON-based request. The call is made with the URL defined in `aspera.conf`. For example:

```
POST URL/validation/files HTTP/1.1
Content-type: application/json
```

The system then generates a JSON accepted or error response (`OK` or `Bad Request`). If a file validation fails, it terminates the session with an error message from the URI.

- **Sample JSON accepted response:** The `"file_encryption"` field is only returned if server-side EAR is present.

```
HTTP 200 OK
{
    "id" : "1111-2222-333",
    "file_encryption" : {
        "passphrase" : "supersecret"
    }
    "aspera_response_object_name" : {
        "startstop" : "start"
        "xfer_id" : "AAAA-BBBB",
        . . .
        "file_csum" : "a1000abf882",
        "file_csum_type" : "sha2-256"
    }
}
```

- **Sample JSON error response:**

```
HTTP 400 Bad Request
{
  "error" : {
    "code" : "1022",
    "message" : "The file fails validation"
  }
}
```

Custom Code for Including and Excluding Files

Administrators can include or exclude files by enabling whitelisting, blacklisting, or another method of their own design. You can do this by creating custom code in the programming language of your choice, using a web server that runs a REST service. (HSTS users have the option to use the web server associated with that installation).

The following is an example of custom code that creates a file blacklist, using a Java servlet deployed on an Apache web server. Note that this code uses the servlet name `SimpleValidator`, which was defined in `web.xml` above.

```
package aspera.validation;

import com.google.gson.Gson;
import com.google.gson.JsonObject;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;

@WebServlet(name = "SimpleValidator")
public class SimpleValidator extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        StringBuilder fileRequestJSON = new StringBuilder();
        BufferedReader reader = request.getReader();
        String line = "";
        Gson gson = new Gson();

        System.out.println("Got Validation request...");
        while (line != null) {
            line = reader.readLine();
            if (!(line == null)) {
                fileRequestJSON.append(line).append("\n");
            }
        }

        ValidationInput validationInput = gson.fromJson(fileRequestJSON.toString(),
ValidationInput.class);

        System.out.println("FileData JSON: " + fileRequestJSON.toString());

        if (validationInput.file != null && validationInput.file.endsWith(".sh")
           || validationInput.file.endsWith(".exe")) {

            JsonObject innerObject = new JsonObject();
            innerObject.addProperty("message", "Cannot transfer executable file!!");
            innerObject.addProperty("code", 1);

            JsonObject jsonObject = new JsonObject();
            jsonObject.add("error", innerObject);

            response.getOutputStream().println(jsonObject.toString());

            response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        }
        else {

            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", true);
            jsonObject.addProperty("data", "File is ok to transfer");
            jsonObject.addProperty("code", 1);
            response.getOutputStream().println(jsonObject.toString());

            response.setStatus(HttpServletResponse.SC_OK);
        }
        return;
    }
}
```

Inline File Validation with Lua Script

To use a Lua script for inline file validation, the administrator creates a base-64 encoded Lua action script and sets the path to that script in the <transfer> section of aspera.conf. During the inline validation, **ascp** automatically generates a request; the parameters for the Lua call are passed to a Lua script defined in aspera.conf.

The parameters for Lua calls are passed to Lua scripts by using the array 'env_table'. The following is an example request body:

```
env_table["startstop"] = "running"
env_table["xfer_id"] = "AAAA-BBBB"
env_table["session_id"] = "1111-2222"
env_table["host"] = "10.0.258.12"
env_table["client_ip"] = "10.0.125.04"
env_table["user"] = "admin"
env_table["userid"] = 24
env_table["direction"] = "send"
env_table["target_rate_kbps"] = 0
env_table["min_rate_kbps"] = 0
env_table["rate_policy"] = "fair"
env_table["cipher"] = "aes-128"
env_table["cookie"] = "xyz"
env_table["manifest_file"] = "/data/manifests/aspera-transfer-1234.txt"
env_table["file"] = "/data/home/luke/test.mpg"
env_table["size"] = 1000000
env_table["start_byte"] = 0
env_table["bytes_written"] = 0
env_table["tags"] = "tags"
env_table["file_name_encoding"] = "utf8"
env_table["file_csum"] = "a1000abf882"
env_table["file_csum_type"] = "sha2-256"
```

Lua Request Body Parameters and Values

Field	Description	Values
"startstop"	Sets the type of validation	start, stop, or running
"xfer_id"	Value used to identify a transfer session	String
"session_id"	Value used to identify a validation session	String
"host"	Server hostname or IP address	Hostname or IP address
"client_ip"	Client IP address	IP address
"user"	SSH account login name	String
"user_id"	Value used to identify the user	String
"direction"	Direction of transfer (send or receive)	send or recv
"target_rate_kbps"	Target rate (in kbps) for file transfer with **ascp**	Integer
"min_rate_kbps"	Minimum rate (in kbps) for file transfer with **ascp**	Integer
"rate_policy"	Defines the **ascp** rate policy. This value is taken from the default configuration in aspera.conf, if not defined here.	fixed, fair, high, or low
"cipher"	The encryption cipher for file data.	String; AES128, ANY, or NONE
"cookie"	The cookie sent to the client system	String

Field	Description	Values
`"manifest_file"`	Path to manifest file, which contains a list of transferred files. The command for this in **ascp** is --file-manifest-path=*file_path*	Filepath
`"file"`	Path to file being validated	Filepath
`"size"`	Allowable file size	Integer (up to 64-bit)
`"start_byte"`		Integer
`"bytes_written"`		Integer
`"tags"`	The JSON request passes the supplied tag values to **ascp**, which in turn passes the tags to the validator.	
`"file_name_encoding"`		String
`"file_csum"`	File checksum	String
`"file_csum_type"`	File checksum type	`md5`, `sha1`, or `any`

The values that are returned from Lua can be used to indicate validation success, validation failure, the script error, or to change the file destination:

Status	Lua return value
Validation success	No value or LRET_OK
Script error	LRET_ERROR followed by an error number or error description string

Lua File Interfaces

Three Lua file interfaces allow Lua scripts to reference files: `lua_stat`, `lua_file_delete`, and `lua_rename`.

- `lua_stat("file_path")`

  Used to gather metadata for the single file specified by *file_path*, where *file_path* is relative to the docroot, if defined. Metadata output include the following:

```
stat_data["exists"] = "true" | "false"
stat_data["size"] = file_size
stat_data["blocks"] = file blocks
stat_data["blocksize"] = block_size
stat_data["type"] = "Invalid" | "S_IFDIR" | "S_IFREG"  | "S_IFCHR" |"S_IFBLK" | "S_IFIFO" |
"S_IFSOCK" |
            "S_IFLNK" | "Block stream" | "Custom" | "Unknown"
stat_data["mode format"] = "Windows format" | "Linux format"
stat_data["mode"] = filemode (format based on mode format above)
stat_data["uid"] = uid
stat_data["gid"] = gid
stat_data["ctime"] = ctime
stat_data["mtime"] = mtime
stat_data["atime"] = atime
```

- `lua_file_delete("file_path")`

  Deletes the single file specified by *file_path*, where *file_path* is relative to the docroot, if defined.

- `lua_rename("old_file_path","new_file_path")`

  Renames the file specified by *old_file_path* with the new name specified by *new_file_path*, both of which are relative to the docroot, if defined.

Lua Logging Interface

You can output simple text strings (format strings are not supported) to the Aspera logs using the **ascp** log interfaces. For example, to log when the Lua script started, enter the following line in a Lua script:

```
lua_log("Lua script started")
```

This produces the following log entry:

```
xxxxxx LOG lua: Lua script started
```

The following **ascp** logging functions are supported:

- `as_log`
- `as_err`
- `as_dbg1`
- `as_dbg2`
- `as_dbg3`
- `as_dbg4`

To use the **ascp** log functions in your Lua script, replace `as` with `lua`.

Miscellaneous Lua Interfaces

- `lua_override_ear_secret("secret")`

  Override the server-side encryption-at-rest (EAR) secret that is set in `aspera.conf` with the specified secret.

Secrets Management with askmscli

The **askmscli** tool can be used to securely manage secrets and keys that can pose a security risk when stored on machines in plain-text format.

The **askmscli** command-line tool provides the following capabilities:

- **Content protection**: Encrypting passphrases used for server-side encryption at rest (SSEAR).
- **Dynamic token encryption key**: The key used for encrypting authorization tokens can be dynamically generated for improved security and time-limited validity.
- **Master key for Redis database**: Encryption of sensitive information in a Redis database, using a secure 256-bit master key set by the system administrator. Sensitive information, for example, could be an access-key-specific content-protection secret or token encryption key.
- **Redis database password**: System administrators can set a secure password for a Redis database to authenticate Redis clients.
- **Stash file protection for keystores**: System administrators can set a stash password to add a stash file-encryption layer to all keystores of a node.

Migration Procedures

The migration steps described below allow you to take full advantage of this security feature. Although not required, Aspera strongly recommends you adopt these measures to ensure the secure usage of Aspera products. The steps consist of running the **askmscli** tool, which stores secrets in file-system-protected keystore files rather than in the `aspera.conf` file.

According to which encryption features you're setting up, the **askmscli** command recognizes three categories of secrets:

`ssear`	Passphrases used for server-side encryption at rest (SSEAR).

`redis-master-key`	Master key for encryption of sensitive data in a Redis database, such as token encryption keys.
`redis-password`	Password for a Redis database to authenticate Redis clients.

The **`askmscli`** tool stores secrets in SQLite DBs. It maintains the encrypted secrets in two databases:

`rootkeystore.db`	Functions as a backup and main source of truth for encrypted secrets.
`localkeystore.db`	Resides in user's home directory (by default). It contains a copy of shared keys like `redis-master-key` and `redis-password` along with a user's own secrets. This allows every user to use the same shared keys without requiring a shared/world-readable database.

The procedures described below are organized to indicate which steps are for upgrades, which are for new installations, and which are for both.

On Linux systems, the Aspera commands in this topic are located in `/opt/aspera/bin`. For your convenience, you may want to add `/opt/aspera/bin` to your PATH environment variable.

Content-Protection Secret

The **`askmscli`** tool sets content-protection secrets only for each user, not for groups and not for all users on a node. Each transfer user requires their own content-protection secret for SSEAR. The steps below describe how to migrate content-protection secrets from `aspera.conf` to a user's local keystore. Two procedures are shown: one for new installations and one for upgrades. Aspera recommends that you do not store content-protection secrets in `aspera.conf`.

For Upgrades:

Before you start, make a backup copy of your `aspera.conf` file.

1. Locate all `content_protection_secret` settings in `aspera.conf`, and make a note of the value that's set for each.
2. Set a content-protection secret for each user:

```
$ echo -n secret | sudo askmscli -u username -s ssear
```

3. Remove all plain-text content-protection secrets from `aspera.conf`. Use these commands:

```
$ sudo asconfigurator -x "set_user_data; user_name,user;
transfer_encryption_content_protection_secret,AS_NULL"
$ sudo asconfigurator -x "set_group_data; group_name,group;
transfer_encryption_content_protection_secret,AS_NULL"
$ sudo asconfigurator -x "set_node_data;
transfer_encryption_content_protection_secret,AS_NULL"
```

For New Installations:

1. Set the content-protection secret for each transfer user by running this command:

```
$ echo -n secret | sudo askmscli -u username -s ssear
```

Token Encryption Key

The following is a brief summary of the steps for encrypting and using dynamic token encryption keys:

(1) Remove existing plain-text token encryption keys from `aspera.conf`.
(2) Set `token_dynamic_key` to `true` in `aspera.conf`.
(3) Set a master key for Redis.

Before you proceed, make a backup copy of your `aspera.conf` file.

For Upgrades and New Installations:

1. Enable the use of dynamic token encryption keys by setting `token_dynamic_key` to `true` in `aspera.conf`.

```
$ sudo asconfigurator -x "set_node_data; token_dynamic_key,true"
```

   NOTE: A dynamic token encryption key can be set for an individual user or a system group.

2. Set a Redis master key using **askmscli**. The master key must be a unique random 256-bit key. The example below uses **openssl** to generate the key. This Redis master key will be used to encrypt the dynamic token encryption key.

```
$ echo -n "`openssl rand -base64 32`" | sudo askmscli -s redis-master-key
```

3. For each transfer user with a token encryption key, run the commands below to initialize the user's keystore:

```
$ sudo askmscli -i -u username
```

4. Set the store for the `asperadaemon` user that runs asperanoded.

```
$ sudo askmscli -i -u asperadaemon
```

5. Restart **asperanoded** to apply the new configuration changes. To test transfers, try an upload and download through your Web application.

   For Upgrades Only:

6. Once all the outstanding tokens created from the old token encryption keys have expired, remove the `token_encryption_key` settings from `aspera.conf`:

```
$ sudo asconfigurator -x "set_user_data; user_name,user; token_encryption_key,AS_NULL"
$ sudo asconfigurator -x "set_group_data; group_name,group; token_encryption_key,AS_NULL"
$ sudo asconfigurator -x "set_node_data; token_encryption_key,AS_NULL"
```

Redis Master Key

System administrators can now set a unique 256-bit Redis master key to encrypt local access-key configuration and dynamic token encryption keys.

For New Installations Only: (Not applicable to upgrades.)

If you have not already created a Redis master key as part of enabling dynamic token encryption keys (as in "Token Encryption Key" above), then:

1. Set a Redis master key using **askmscli**. The master key must be a unique, random 256-bit key. The example below uses **openssl** to generate the key. This Redis master key will be used to encrypt both the dynamic token encryption key and access keys in Redis.

```
$ echo -n "`openssl rand -base64 32`" | sudo askmscli -s redis-master-key
```

2. For the transfer user associated with the an access key, run **askmscli** to initialize the user's keystore (if not done previously):

```
$ sudo askmscli -i -u username
```

3. If you are using your server as a *tethered node* with Aspera on Cloud, or if you are using the access key authentication feature, then you must encrypt your access key.

   a. List your node user and password with the command:

```
#/opt/aspera/bin/asnodeadmin -l
```

   b. Retrieve your access key with this command syntax:

```
curl -kiu node_user:password https://localhost:9092/access_keys
```

(For more information, see Node API Reference.)

c. Encrypt your access key data with this command syntax:

```
asnodeadmin --encrypt-access-key --access-key access_key
```

Redis Password

System administrators can set a secure password for clients to authenticate with a Redis database. When the authorization layer is enabled, Redis refuses any query by unauthenticated clients. A client can authenticate itself by sending the **AUTH** command followed by the password.

For New Installations Only: (Not applicable to upgrades.)

1. Set a Redis password:

```
$ export redis_pass="password"
$ echo -n $redis_pass | sudo askmscli -s redis-password
```

2. For transfer users who will query Redis using the `asnodeadmin` tool, run **`askmscli`** to initialize the user's keystore with the new Redis password:

```
$ sudo askmscli -i -u username
```

NOTE: Running **`askmscli`** to initialize an existing user keystore also updates the keystore with the new shared secret (`redis-password` in this case)

3. In order for the Redis password to persist across reboots, perform the following steps:

a. Temporarily change the ownership of the Redis configuration file `aspera_31415.conf` to the user `asperadaemon`.

```
$ chown asperadaemon /opt/aspera/etc/redis/aspera_31415.conf
```

b. Update the configuration file to save the password across reboots:

```
$ asredis -p 31415
127.0.0.1:31415> CONFIG SET REQUIREPASS redis_passwd
OK
127.0.0.1:31415> AUTH redis_passwd
OK
127.0.0.1:31415> CONFIG REWRITE
OK
127.0.0.1:31415> quit
```

c. Restore `aspera_31415.conf` ownership to `root`

```
$ chown root /opt/aspera/etc/redis/aspera_31415.conf
```

.

Stash File Protection for Keystores

You can set a stash password to add a stash file-encryption layer to all keystores of a node:

- Use the `askmscli` tool to set a stash password for the node.
- The stash password is used to derive a unique key for every keystore.
- The unique keystore key is then used to encrypt and decrypt the keystore secrets.

The **`askmscli`** tool can be used to set a stash password using either standard input, or a file. For example, using standard input:

```
$ echo -n "r@nd0ms3curepassw0rd" | sudo /opt/aspera/bin/askmscli -P-
Stash password set successfully
```

Or, using a file that contains a stash password:

```
$ echo -n "r@nd0ms3curepassw0rd" >> /tmp/random-stash-password.txt
$ sudo /opt/aspera/bin/askmscli -P /tmp/random-stash-password.txt
Stash password set successfully
```

Note: Setting a stash password automatically upgrades the `rootkeystore.db` (or creates one if it does not exist).

After setting a stash password, re-initialize any existing user keystores (using the **`-i`** option) so that encryption uses the keystore key.

```
$ sudo /opt/aspera/bin/askmscli -i -u $USER
Keystore initialized successfully
```

All secrets added after re-initializing the existing keystores are automatically encrypted using the keystore key.

The limitations on using stash-file protection are:

- Once a stash password is set, and user keystores are initialized, deleting a stash file will make all keystore records inaccessible (root and user).
- Changing or rotating a stash password is not currently supported.

ascp: Transferring from the Command Line

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 136
- "Ascp File Manipulation Examples" on page 139

Another command-line FASP transfer program, 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 Ascp4" on page 162 and "Comparison of Ascp and Ascp4 Options" on page 157.

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.

The *growing files* feature can be used with the *source* option to start transferring files to the target directory while they are still being written to the source directory.

Note: To use the growing files feature, the source file must be on a native file system. Growing files cannot be larger than 8 exabyte - 1 (9,223,372,036,854,775,807 bytes). However, the maximum file size of the file system will override a setting that is larger.

Growing files use can also be configured statically with `aspera.conf`, see "aspera.conf - File System Configuration" on page 91. See also "Ascp General Examples" on page 136.

To use the growing files feature with **ascp**, the *source* parameter can be used with the following syntax:

```
source?grow=wait_time[&wait_start=[mtime | null_read]][[&confirm_stop=[ true | false ]]
```

A file transfer is deemed to be complete when the time since last update to the source file reaches the *wait_time* value (in seconds). However, the time is only sampled when all currently available source data has been transferred. In other words, if more data arrives after the wait time has elapsed, but the transfer is still in progress, the additional data will still be transferred.

grow

Enables the growing file feature and is used to set the wait time in seconds. The wait time is the amount of time that is allowed to pass before a file that is not changing is treated as complete. The `grow` element must be set to a non-negative integer to define wait time. If it is set to a non-numeric string, the default wait time of 10 seconds is used.

wait_start

Can be used to specify how time is calculated to determine if a file is complete. If `mtime` is used, then the file modification time is used when calculating the wait time. If `null_read` is used, then the time of a file read that returns zero bytes is used. The default is `mtime`.

confirm_stop

Can be used to indicate when all the data has been added to the source file and to prevent any additional wait time following a read of EOF.

Note that `confirm_stop` is ignored if `wait_start` is set to `null_read`.

To use `confirm_stop`, set it to `true` (the default is `false`). Then use an external program to adjust the source file mtime upon completion of writing data to the source file, using the following criteria:

```
mtime < current_system_time - wait_time, mtime != 0
```

Any value for mtime that meets this criteria is acceptable to flag this condition except mtime = 0,which is used to flag a file error. You can, for example, use **`touch 1`**. If mtime is not adjusted before the timeout is reached, an error will be generated.

An alternative method for defining the `wait_time` value is to use modifiers for powers of 1,024. However, the value must be less than 8 * 2^60. The modifier must consist of an integer, and a unit specifier. The unit specifiers are:

- k or K for 1 * 1024
- m or M for 1 * 1024 * 1024
- g or G for 1 * 1024 * 1024 * 1024

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 354. For examples, see "Ascp General Examples" on page 136.

Websocket Protocol

The Websocket protocol can be used instead of SSH or HTTPS for client connections with the transfer server. In order to use it, you must configure `aspera.conf` specifically for Websocket use. Then for transfers, you must use the **ascp `--ws-connect`** option to specify using Websocket, and the **`-P`** option to specify the Websocket port (9093).

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 "Multi-Session Transfers" on page 140.

-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 explicitly 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 "Securing Your SSH Server" on page 14.

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 145.

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 .

--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 “Reporting Checksums” on page 55 .

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 156.

--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 136.

-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*** **|** ***cert_file***
The **`-i`** option can be used to specify either:

- an SSH private key file, for authenticating a 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.
- a Certificate Authority certificate, for use with fallback transfers or with Websocket use, when you do not want to use the system default certificate.

-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 139.

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 "Multi-Session Transfers" on page 140.

-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 145.

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* | *websockets-port*
Use the specified TCP port to initiate the FASP transfer session, using the port number that is appropriate for your use of either SSH or Websocket.

-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.

--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.)

--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 136.

-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.

-SSH
Use an external SSH program instead of the built-in libssh2 implementation to establish the connection with the remote host. The desired SSH program must be defined in the environment's PATH variable. To enable debugging of the SSH process, use the `-DD` and `--ssh-arg=-vv` options with **ascp**.

--ssh-arg=*ARG*
Add *ARG* to the command-line arguments passed to the external SSH program (this implies using -SSH). This option may be repeated as needed to supply multiple separate SSH arguments. The order is preserved. The *ARG* elements are inserted before any key file(s) supplied to **ascp**, and before the user/host argument.

--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 139.

--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 150. 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 for Lua scripts that can be run with transfer events. See "Transfer Session Data Accessible to Scripts" on page 179.

--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 356 .

-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.

--ws-connect
Use Websocket instead of SSH for client connections with the transfer server.

-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 automated execution of Lua scripts ("Automated Execution of Lua Scripts with Transfer Events" on page 177).

Options:

`-i` *cert_file*
By default **ascp** uses the system certificates. However, the **`-i`** option can be used to use the specified Certificate Authority certificate for fallback transfers, and for Websocket.

`-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.

About this task

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

- **Growing Files**

  The growing files feature allows you to start transferring files to the target directory while they are still being written to the source directory.

  Download the growing file `myfile` with a wait period of 120 seconds, and using a zero-byte read when calculating the wait time.

```
ascp --mode=recv --user=root --host=10.0.0.2 "file:////tmp/myfile?
grow=120&wait_start=null_read" file:////tmp2/mylocalfile
```

  To support this command, the `ascp.conf` file would have to include the following configuration:

```
<default>
    <file_system>
        <access>
            <paths><path><absolute>
                file:////tmp?grow=120;wait_start=null_read
            </absolute></path></paths>
        </access>
    </file_system>
</default>
```

For more information, see the discussion of `ascp.conf` configuration for growing files in "aspera.conf - File System Configuration" on page 91.

- **Using the Websocket Protocol**

  This example shows how to use the Websocket protocol for a transfer. The Aspera Node Service provides a Websocket server, which must be enabled (see "aspera.conf - Websocket Configuration" on page 67). Because the Ascp client only supports a secure Websocket transfer (HTTPS), the Aspera Node Service must be configured for HTTPS, or must use a reverse proxy to terminate the secure connection.

  A basic token, bearer token or transfer token must be used with a Websocket connection. For information about tokens, see "Introduction to Aspera Authentication and Authorization" on page 309.

  The following **ascp** options are required for using Websocket:

  --ws-connect
  Specifies using Websocket.

  -P
  Specifies the Websocket port (9093).

```
# ascp -L- --ws-connect -P 9093  --host=www.example.com --mode=send --user=xeno c:/Users/xeno/
Desktop/myfile  /Desktop/ dest
```

- **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`:

- **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):

  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:

- **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 354.

- **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/folderB/file3`	`/in/out/folderA/file2`
`/clips/out/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
  ```

Multi-Session Transfers

Ascp can transfer content faster by using multi-session transfers (also known as parallel transfers and multi-part transfers) to and from multi-node and multi-core servers and clusters, on premises or in the cloud. This article describes the syntax of a multi-session transfer and provides an example.

Multi-session syntax

To run simultaneous **ascp** transfers, you can run each command from its own terminal window, run a script in a single terminal, or background processes with the shell.

For a typical push (`--mode=send`) transfer:

```
# ascp -C nid_1:ncount -l max_rate [-O port_1][--multi-session-threashold=threshold] [--
tags={\"aspera\":{\"xfer_id\":\"transfer_id\"}}] source_path hostname:destination_path
# ascp -C nid_2:ncount -l max_rate [-O port_2] [--multi-session-threashold=threshold] [--
tags={\"aspera\":{\"xfer_id\":\"transfer_id\"}}] source_path hostname:destination_path
...
# ascp -C nid_n:ncount -l max_rate [-O port_n] [--multi-session-threashold=threshold] [--
tags={\"aspera\":{\"xfer_id\":\"transfer_id\"}}] source_path hostname:destination_path
```

Where:

- `-C` *`nid:ncount`* tells Ascp that the same source and destination are being used by multiple, concurrent sessions. *nid* is the node ID and *ncount* is the number of nodes or cores. The valid range of values for *nid* and *ncount* is 1 through 128, and *nid* must be less than or equal to *ncount*.
- `-O` *`port`* is used to assign each session to a different UDP port. This is **required** when the server's operating system does not support concurrent sessions using the same UDP port. This applies to Windows, MacOS, and Solaris operating systems.

  Note: Make sure that the server's firewall is configured to accept transfers on the range of UDP ports.
- `--multi-session-threshold` is an optional argument that enables files to be split between sessions. The threshold value specifies, in bytes, the smallest-size file that can be split. Files greater than or equal to the threshold are split, while those smaller than the threshold are not. If the multi-session threshold is set to 0 (zero), files are not split.
- `--tags={\"aspera\":{\"xfer_id\":\"`*`transfer_id`*`\"}}` is **required** for multi-session transfers to cloud in order to provide a transfer ID. The transfer ID is the same for all the sessions in the multi-session transfer. If an upload is restarted with the same *xfer_id* then the transfer is resumed, but if a different *xfer_id* is used then the upload is completely restarted.
- If you are uploading to a cloud-based AWS S3 cluster, you must authenticate with an access key or Assumed role rather than an IAM role.
- If you are self-managing an Aspera server or cluster of Aspera servers in the cloud (you installed IBM Aspera High-Speed Transfer Server on a VM), you must configure the server for multi-session transfers.

File-spliting with multi-session threshold

The value of the multi-session threshold depends on the target rates that a single **ascp** transfer can achieve on your system for files of a given size, as well as the typical distribution of file sizes in the transfer list.

Note: A default value for the threshold can be specified in the server and client `aspera.conf` by setting `<multi-session_threshold_default>` in the `<default>` section. The command-line setting overrides the `aspera.conf` setting. If the client's `aspera.conf` does not specify a default value for the threshold, then the server's setting is used (if specified). If neither the client nor the server set a multi-session threshold, then no files are split.

To set a value (in bytes) from the command line, run the following:

```
# asconfigurator -x "set_node_data;transfer_multi_session_threshold_default,threshold"
```

Multi-Session Transfer Example

The following example shows a multi-session transfer on a dual-core system. Together, the two sessions can transfer at up to 2 Gbps and each session uses a different UDP port. No multi-session threshold is specified on the command line or in `aspera.conf`, so no file splitting occurs.

```
# ascp -C 1:2 -O 33001 -l 1000m /dir01 10.0.0.2:/remote_dir
# ascp -C 2:2 -O 33002 -l 1000m /dir01 10.0.0.2:/remote_dir
```

If `dir01` contains multiple files, **ascp** distributes the files between each command to get the most efficient throughput. If `dir01` contains only one file, only one of the commands transfers the file.

In the following example, the multi-session threshold is used to enable file splitting:

```
# ascp -C 1:2 -O 33001 -l 100m --multi-session-threshold=5242880 /dir01 10.0.0.2:/remote_dir
# ascp -C 2:2 -O 33002 -l 100m --multi-session-threshold=5242880 /dir01 10.0.0.2:/remote_dir
```

In this case, if `dir01` contains multiple files, all files less than 5 MB are distributed between sessions, while all files 5 MB or larger are split and then distributed between sessions. If `dir01` contains only one file and that file is 5 MB or larger, then the file is split, otherwise the file is transferred by one session.

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.

Note: Do not use zero-byte files with standard I/O transfers.

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.

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

- 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. 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'`.

Example	Transfer Result
	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

Filters can be specified on the **ascp** command line and in `aspera.conf`. Ascp applies filtering rules that are set in `aspera.conf` *before* it applies rules on the command line.

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
[!...]	[!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

You can use this procedure to test your filtering rules.

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 a server. For example:

   ```
   # ascp /tmp/src my_user_name@my_demo.example.com:Upload/
   ```

   Where the user is "my_user_name", and the target is the Upload directory.

   At the prompt, enter my_user_name's password.
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' my_user_name@my_demo.example.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 my_demo.example.com to /tmp/dest:

```
# ascp rules aspera@my_demo.example.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: Ascp4 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
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

	Server setting = create, follow (default)	Server setting = create	Server setting = follow	Server setting = follow_wide	Server setting = none
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"
```

For more information, see “aspera.conf - File System Configuration” on page 91.

Client Configuration

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

Creating SSH Keys

About this task

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.

If you are using this machine as a client to connect to other Aspera servers with public key authentication, you need to generate a key-pair for the selected user account, as follows:

Procedure

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. Generate an SSH key-pair.

   In the `.ssh` folder, use the **ssh-keygen** command to create a key pair.

   ```
   # ssh-keygen -m key_format -t key_type
   ```

   - For *key_format*, specify a format that is supported by the SSH server.
   - For *key_type*, specify either RSA (`rsa`) or ECDSA (`ecdsa`).

   At the prompt that appears 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.

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 pkhsts8 -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. The instructions for installing the public key on the server can be found in the "Setting Up a User's Public Key on the Server" on page 24; however, the server may be installed on an operating system that is different from the one where your client has been installed.

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 HSTS or HSTE 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 Processing Script

An alternative to enabling and configuring the file manifest to collect checksum reporting is to set up a script to report the values. See “Automated Execution of Lua Scripts with Transfer Events” on page 177.

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 Lua transfer event 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 136.

Note: When a transfer to HSTS 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. HSTS 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 Ascp4 Options

Many command-line options are the same for Ascp and Ascp4; 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 Ascp4, 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	Ascp4
-6	
-@[*range_low*:*range_high*]	
-A, --version	-A, --version
--apply-local-docroot	
-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	--dest64
-E *pattern*	-E *pattern*
-e *prepost_filepath*	
	--exclude-newer-than=*mtime*
	--exclude-older-than=*mtime*
-f *config_file*	-f *config_file*
	--faspmgr-io
--file-checksum=*hash*	
--file-list=*filepath***	--file-list=*filepath***
--file-pair-list=*filepath*	

Ascp	Ascp4
-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*]
-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*	--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*	

Ascp	Ascp4
-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-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.

Procedure

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 150.

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

Ascp4 is a FASP transfer binary similar to Ascp but it has different strengths as well as capabilities that are unavailable with Ascp.

Introduction to Ascp4

Ascp4 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 Ascp4 and Ascp are automatically installed with IBM Aspera High-Speed Transfer Server, IBM Aspera High-Speed Transfer Endpoint, and IBM Aspera Desktop Client.

Ascp4 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 Ascp4 Options” on page 157. 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. Ascp4 currently supports transfer tokens, which must be created by using **astokengen** with the **--full-paths** option. For more information, see “Transfer Token Generation (astokengen)” on page 313.

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.

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.

Ascp4 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 "Securing Your SSH Server" on page 14.

--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`.

--dest64
Indicate that the destination path or URI is base64 encoded.

-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 145.

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.

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

--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-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 156.

--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.

--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`)

-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.

For streaming, the value should be equal to or greater than the bitrate of the video.

-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 145.

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.

`--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={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.

`--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 (`ascp4` or other) might access the source directory at the same time.

`--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.

-SSH
Use an external SSH program instead of the built-in libssh2 implementation to establish the connection with the remote host. The desired SSH program must be defined in the environment's PATH variable. To enable debugging of the SSH process, use the `-DD` and `--ssh-arg=-vv` options with **ascp4**.

--ssh-arg=*ARG*
Add *ARG* to the command-line arguments passed to the external SSH program (this implies using-SSH). This option may be repeated as needed to supply multiple separate SSH arguments. The order is preserved. The *ARG* elements are inserted before any key file(s) supplied to **ascp4**, and before the user/host argument.

--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 139.

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 150. 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 Lua scripts that can be run with transfer events. See "Transfer Session Data Accessible to Scripts" on page 179.

--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".

Ascp4 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.

About this task

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

Procedure

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`.

Ascp4 Examples

The command options for **ascp4** are generally similar to those for **ascp**. The following examples demonstrate options that are unique to Ascp4. These options enable reading management commands, transfer TCP and UDP data streams, and enable read/write concurrency.

For Ascp examples, see “Ascp Command Reference” on page 121. See “Comparison of Ascp and Ascp4 Options” on page 157 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
```

- **Streaming**

  See “Ascp4 Video Streaming Examples” on page 174.

- **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
```

Built-in I/O Providers

Input/Output providers are library modules that abstract I/O scheme in Ascp4 architecture. Ascp4 has the following built-in I/O providers:

- file (as a simple path or `file://`*`path`*)
- TCP (as `tcp://192.168.120.11`)
- UDP (as `udp://233.3.3.3`)

File provider

The local disk can be specified for **ascp4** I/O by using a simple path or URL that starts with `file`. The following paths identify the same file (`/test/ascp4.log`) on the disk:

```
file:///test/ascp4.log
/test/ascp4.log
file://localhost:/test/ascp4.log
```

Similarly, the following URLs identify the same file (`test/ascp4.log`) on the disk:

```
file:///test/ascp4.log
test/ascp4.log
```

TCP provider

A TCP data stream can be used for **ascp4** I/O by specifying a URL that starts with `tcp`. **ascp4** reads TCP data from the source and writes TCP data on the destination. Use the following format to specify a TCP provider on the source or destination:

`tcp://`*`ip_address`*`:`*`port`*`[?`*`option=value`*`[&`*`option=value`*`]]`

The TCP provider of the sender can also be specified with the following format:

`tcp://:`*`port`*`[?`*`option=value`*`[&`*`option=value`*`]]`

With this format, **ascp4** listens on the specified port up to a specified time (`maxidle`, see the following description of options for TCP provider URLs).

The TCP provider URL accepts the following options:

`port=`*`N`* — Set the network port number, default: 0.
`iosize=`*`N`* — Specify the read/write size, default: 32 KB.
`maxsize=`*`N`* — Set the maximum stream length, in bytes, no default.

`maxtime=N` — Set the maximum stream duration, in seconds, no default.
`maxidle=N` — Set the maximum idle duration, in seconds, default: 10 sec.
`rcvbufsz=N` — Set the receive buffer size, default: 4 MB.
`sndbufsz=N` — Set the send buffer size, default: 4 MB.
`ifaddr=ip_address` — Specify the TCP connection interface address.
`srcaddr=ip_address` — Specify the TCP connection source-specific address.

UDP provider

A UDP data stream can be specified for **ascp4** I/O by using a URL that starts with udp. If the UDP stream is a multicast IP address, then **ascp4** connects to the multicast address. **ascp4** reads the UDP datagrams on the source and writes UDP datagrams on the destination. A UDP-provider filepath has the following format:

`udp://ip_address:port[?option=value[&option=value]]`

The UDP provider URL accepts the following options:

`pktbatch={0|1}` — Enable packet batching in read/write. Default: 1.
`maxsize=N` — Set the maximum stream length. Default: unlimited.
`maxtime=N` — Set the maximum stream duration, in seconds. Default: unlimited.
`maxidle=N` — Set the maximum idle duration, in seconds. Default: unlimited.
`rcvbufsz=N` — Set the receive buffer size. Default: 10 MB.
`sndbufsz=N` — Set the send buffer size. Default: 10 MB.
`ifaddr=ip_address` — Set the multicast interface. Default: 0.0.0.0.
`srcaddr=ip_address` — Set the multicast source for IGMPb3 source-specific multicast.
`ttl=N` — Set the multicast time-to-live. Default: 1.
`loopback=N` — Set the multicast loopback. Default: 1.
`dontfrag=N` — Prevent fragmentation of outgoing packets. Default: 0.

ascp4: Streaming

Aspera streaming uses the FASPStream technology to send unicast or multicast video stream over the internet, using **ascp4**. Streaming is generally used to transfer a video stream from a *stream provider*, a system external to Aspera streaming that produces a video stream, to a *stream consumer*, a system external to Streaming that receives a video stream.

Note: Streaming capabilities are only provided with the IBM Aspera Enterprise and IBM Aspera Endpoint products. Streaming is not provided with, or supported by, the HSTS or HSTE standalone products.

The most common example of an input stream is media, which is encoded as a *transport stream* (often referred to as MPEG-TS). On the source host, the UDP provider captures packets delivered to a local multicast or unicast UDP port. The stream is transported reliably and in order to the remote host, where it is delivered to the specified local multicast or unicast UDP port. Video streaming that uses TCP and file I/O is also supported. Streams are reliably and bit-perfect replicated at the receiving endpoint with minimal and predictable latency, with rates from 10s of Mbps to multiple Gbps, and follow the standard Aspera FASP security framework.

Streaming can run on either a standalone computer or as an embedded service on third-party devices (for example, a video encoder). In a single point-to-point transfer, the *sender* is the system that initiates the transfer and sends a request to a *receiver* for a streaming session. In general, a *sender* may only produce a stream and a *receiver* may only consume a stream. Additionally, a *server* may act as both a *sender* and a *receiver* and is not limited to a number of concurrent streams up to the bandwidth limit purchased.

Using Ascp4 for Streaming Video

Ascp4 can be used for streaming as well as regular file transfers. The syntax is similart to a regular **ascp4** file transfer, but the source and destination are URI paths.

Required Configuration for Multicast-to-Multicast Transfers

The transfer user who authenticates the multicast-to-multicast video stream transfer must have no docroot configured in `aspera.conf`. A file restriction can be set instead to restrict access.

Run the following command to unset a docroot and set a file restriction:

```
# asconfigurator -x "set_user_data;user_name,username;absolute,AS_NULL;file_restriction,|
restriction"
```

The restriction can be set to allow all access (*) or limited by protocol, hostname or path:

Restriction	Format Example
By protocol	udp://* tcp://*
By protocol and hostname	udp://hostname*
By protocol, hostname, and port	tcp://hostname:5000*

Basic Ascp4 Usage for Streaming

```
# ascp4 -l target_rate --mode=mode --host=remote_hostname --compression=none --user=username --
read-threads=1 --write-threads=1 input_uri output_uri
```

- The **ascp4** `-l` target rate option should be sent to a value that is equal to or greater than the bitrate of the video.
- **ascp4** streaming supports two transfer directions: `send` and `recv`.
- The **ascp4** command defaults to multiple threads, but for reliable and in-order transport of streams you must use only one read and write thread by specifying `--read-threads=1 --write-threads=1`.
- The video stream source and destination can be `udp://`, `tcp://`, or `file://`. For more information, see “Built-in I/O Providers” on page 171

For command line usage examples, see “Ascp4 Video Streaming Examples” on page 174.

For detailed information about **ascp4** options, see “Ascp4 Command Reference” on page 162.

Multicast URI Syntax

The input multicast URI and the output multicast URI uses the same syntax.

```
multicast_protocol_scheme://stream_ip_address:port?option=value&option=value...
```

The multicast protocol scheme can be either udp or `mcast`. If the IP address of your video stream is a multicast address, **ascp4** uses multicast regardless of the protocol scheme (in other words, both udp and `mcast` use multicast). In order to use unicast addresses, you must use the udp scheme.

You can configure properties of the stream by adding options to the URI after the question mark (?), each separated by an ampersand (&). The following table describes the supported options.

Option	Description	Default
`pktbatch={1|0}`	How to handle packet read and write. If 1, batch read and write UDP datagrams. If 0, read and write one packet at a time.	1

Option	Description	Default
batchmicros=*N*	Timeout for batching, in microseconds.	No default
maxsize=*maximum_size*	Maximum stream length	No default
maxtime=*maximum_time*	Maximum stream duration, in seconds	No default
maxidle=*maximum_time*	Maximum idle duration, in seconds	No default
rcvbufsz=*buffer_size*	Receive buffer size	10MB
sndbufsz=*buffer_size*	Send buffer size	10MB
ifaddr=*ip_address*	Multicast interface IP address	0.0.0.0
srcaddr=*ip_address*	Multicast source IP address	0.0.0.0
ttl=*hops*	Multicast time-to-live	1
loopback=*boolean*	Multicast loopback	1

Ascp4 Video Streaming Examples

Use the following examples as a guide for creating your own streaming transfers with **ascp4**.

- Send a multicast stream:

```
# ascp4 --mode=send --host=desthost --compression=none --read-threads=1 --write-threads=1
udp://233.3.3.3:3000?loopback=1&ttl=2 udp://233.4.4.4:3000?loopback=1&ttl=2
```

- Capture a local multicast stream and send it to the receiver as a UDP unicast stream:

```
# ascp4 --mode=send --host=desthost --compression=none --read-threads=1 --write-threads=1
udp://233.3.3.3:3000?loopback=1&ttl=2 udp://localhost:3000/
```

- Read a TCP stream from 192.168.10.10 port 2000 and send it to 10.10.0.51. On 10.10.0.51, write the stream to localhost port 3000.

```
# ascp4 -l 6000 --host=10.10.0.51 --mode=send --read-threads=1 --write-threads=1 tcp://
192.168.10.10:2000 tcp://localhost:3000
```

- Send a multicast UDP stream on 233.3.3.3 port 3000 to host 192.168.0.11, then multicast the stream on 233.3.3.3 port 3001.

```
# ascp4 -l 6000 --host=192.168.0.11 --mode=send --read-threads=1 --write-threads=1
    udp://233.3.3.3:3000/?pktbatch=0 udp://233.3.3.3:3001/?loopback=1
```

- Multicast using the same multicast IP address and varying the multicast port.

```
# ascp4 -L/opt/test-local-01 -R/opt/test-remote-01 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.1:3001?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.1:4001?rcvbufsz=100M&loopback=0"
# ascp4 -L/opt/test-local-02 -R/opt/test-remote-02 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.1:3002?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.1:4002?rcvbufsz=100M&loopback=0"
# ascp4 -L/opt/test-local-03 -R/opt/test-remote-03 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.1:3003?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.1:4003?rcvbufsz=100M&loopback=0"
# ascp4 -L/opt/test-local-04 -R/opt/test-remote-04 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.1:3004?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.1:4004?rcvbufsz=100M&loopback=0"
```

- Multicast using the same multicast port and varying the multicast IP address:

```
# ascp4 -L/opt/test-local-01 -R/opt/test-remote-01 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.1:3001?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.1:4001?rcvbufsz=100M&loopback=0"
# ascp4 -L/opt/test-local-02 -R/opt/test-remote-02 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.2:3001?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.2:4001?rcvbufsz=100M&loopback=0"
```

```
# ascp4 -L/opt/test-local-03 -R/opt/test-remote-03 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.3:3001?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.3:4001?rcvbufsz=100M&loopback=0"
# ascp4 -L/opt/test-local-04 -R/opt/test-remote-04 -DD -l 15m --mode send --host 10.132.117.2
--user root --read-threads 1 --write-threads 1 --compression none "udp://233.33.3.4:3001?
sndbufsz=100MB&ifaddr=10.131.117.1" "udp://233.44.4.4:4001?rcvbufsz=100M&loopback=0"
```

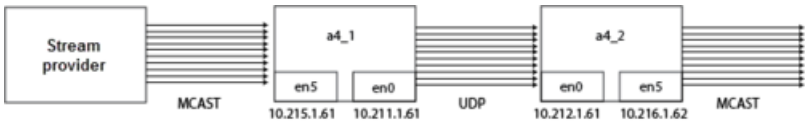
Configuring macOS Server for Multicast Streams

If you are sending or receiving multicast streams from a macOS server, multicast fails if the receiving or sending interface is not the macOS default interface. If no default gateway is defined on a macOS server, the default interface is **en0**. To use a different interface than the default, change the default interface for your server

Run the following commands:

```
# route delete default new_interface_ip
# route add default new_interface_ip
```

Example



Server Name	en0 IP Address	en5 IP Address
a4_1	10.211.1.61	10.215.1.61
a4_2	10.212.1.62	10.216.1.62

In this example, server **a4_1** acts as the multicast receiver and sends the stream over UDP to server **a4_2**. Server **a4_2** then broadcasts the multicast streams to waiting receivers. Since both servers are using a non-default interface (not **en0**) to receive and send the multicast streams, we must set the interface receiving and sending the multicast streams as the server default interface.

Run the following command on **a4_1**:

```
# route delete default 10.215.0.1
# route add default 10.215.0.1
```

Run the following command on **a4_2**:

```
# route delete default 10.216.0.1
# route add default 10.216.0.1
```

Note: The interface IP addresses `10.215.0.1` and `10.216.0.1` routes are the primary gateways and route all `10.215.0.*` and `10.216.0.*` traffic, respectively.

Troubleshooting Stream Transfers

Multicast Transfer Fails with "Error: Empty file list from file/stdin"

This error might indicate that the transfer user has a docroot set in `aspera.conf`, which is not supported for multicast-to-multicast transfers. See “Using Ascp4 for Streaming Video ” on page 173.

Transfer from macOS Fails

To diagnose the problem, re-run the stream transfer with the logging level set to debug by adding `-DD` to the command. After the transfer fails, open the log file (*homedir*`/Library/Logs/Aspera/`) and search for an ERR response.

ERR udp_io_open: failed to set rcvbugsz=10485760 (e=55)(ENOBUFS)

This error indicates that the socket buffer size on the Mac computer is too small to send or receive UDP packets. To increase the socket buffer size, specify a large buffer size by adding the following parameters to the URL:

```
url/?rcvbufsz=4000000&sndbufsz=4000000
```

With these parameters, a transfer to and from a Mac computer is written similar to the following example:

```
# ascp4 -DD -l 15m --mode send --host 10.13.117.12 --user root --read-threads 1 --write-threads
1 --compression none "udp://233.13.13.2:3002/?rcvbufsz=4000000&sndbufsz=4000000" "udp://
233.14.14.2:4002/?rcvbufsz=4000000&sndbufsz=4000000"
```

Testing a Video Stream

You can test a video stream by setting up an IBM Aspera Streaming connection between two computers, start a multicast with an encoded transport stream, test that the receiving computer is receiving packets, and play the streamed media content.

About this task

The following instructions require two computers installed with Streaming:

- computer A: Linux computer with the built-in sender license.
- computer B: Linux computer with a receiver license.

Procedure

The following steps must be performed on computer A:

1. Start **ascp4** to transport streams when an input is available.
   The following example assumes you have SSH key access to computer B from computer A.

   ```
   # ascp4 --mode=send --user=computerB_user -i ~/.ssh/id_rsa --host=computerB  --
   compression=none --read-threads=1 --write-threads=1 udp://233.3.3.3:3000?loopback=1&ttl=2
   udp://233.4.4.4:4000?loopback=1&ttl=2
   ```

   Note: For more information about the command syntax for starting a FASPStream, see .
2. Download a test file to stream.

   In your browser, download the ed24p_00.zip test file from the www.w6rz.net website (a community transport stream testing website):

   http://www.w6rz.net/ed24p_00.zip

   Extract the contents into an easily accessible folder. You may provide your own media file, but the examples in this documentation assume that you are using the ed24p_00.zip transport stream file located at /temp/ed24p_00.ts.
3. Provide a multicast stream of a test file to **ascp4**.
   Run the **ffmpeg** command with the location of the media file and set the URI of the resulting stream:

   ```
   # ffmpeg -re  -i /temp/ed24p_00.ts -vcodec copy -acodec copy -f mpegts  "udp://
   233.3.3.3:3000?ttl=2&pkt_size=1316"
   ```
4. Check to see the output of **ascp4** to make sure the Rate of transfer is going up to the expected speed.

Now that your stream is running and **ascp4** shows that it is transporting the stream, check the receiver is receiving the media file. The following steps must be performed on computer B:

5. Run the **tcpdump** command to check streams are coming.

The port number corresponds to port configured in the destination multicast URI. In the example below, the destination port was configured as 4000.

```
# tcpdump upd and port 4000
```

6. Play the media file over the stream.

   The following example uses the third-party, open-source **vlc** command. If you do not have **vlc** on your computer,

   Using VLC, play the media from the stream.

   a) Open VLC.

   b) Click **Open media**. In the resulting dialog, go to the **Network** tab and click **Open RTP/UDP Stream**.

   c) Configure the settings according to your multicast URI.

Option	Value
Protocol	UDP
Mode	Multicast
IP Address	233.4.4.4
Port (for the IP Address)	4000

Results

Your media file should now be playing in the VLC media player.

Automated Execution of Lua Scripts with Transfer Events

Lua is embedded in Ascp and Ascp4, and HSTS can be configured to execute custom Lua scripts automatically at specific transfer events. In addition, Lua functions that can be used for monitoring and managing transfer sessions are provided by Ascp and Ascp4.

The transfer events at which scripts can be executed are:

- Session Start
- Session Stop
- Session Progress
- File Start
- File Stop

Session Progress refers to the periodic event that generates the stats report found in the management stream.

To execute Lua scripts automatically, `aspera.conf` must include a `<transfer>` element, with sub-elements that identify the scripts and the transfer events at which they should be run. Scripts can be identified by pathname, or they can be embedded in the sub-element itself in the form of base64 output. Scripts can access transfer session data, which is stored in a Lua table (`env_table[]`). They can also make use of several functions that are provided to facilitate file system operations, logging, and aborting a session.

Configuration for Lua Script Execution

You must configure the transfer server using the `aspera.conf` file in order execute Lua scripts automatically.

First you add a `<transfer>` element to `aspera.conf`. Then, within the `<transfer>` element, you add sub-elements to identify the transfer events and the scripts. The scripts can be identified by pathname, or the base64 encoded version of the scripts (or path to the scripts) can be used directly.

Scripts can access transfer session data, which is stored in a Lua table (`env_table[]`). They can also make use of several functions that are provided to facilitate filesystem operations, logging, and aborting a session.

Event Types

Scripts can be automatically executed at each of these events:

- session start
- session stop
- session progress
- file start
- file stop

Note: *Session progress* reporting is initiated when a file transmission starts. Transfer status is then updated every second during transmission.

Event Type and Script Specification

Within the `<transfer>` element, you use sub-elements that identify the type of event and the script that will be run at that event. You can identify the script by its pathname, or you can use the base64 encoding of the script itself. A script can be used for more than one event, whether referenced by its filepath or included base64 encoded format.

To simplify developing your script, you may find it useful to temporarily specify the script by pathname as well as use base64 encoding of the script for a given execution event (in separate specification lines).

If both filepath and base64 specifications are used, the pathname-based specification will override a base64 specification.

Session Start
<lua_session_start_script_base64></*my_lua_session_start_script_base64*>
<lua_session_start_script_path></*my_lua_session_start_script_path*>

Session Stop
<lua_session_stop_script_base64></*my_lua_session_stop_script_base64*>
<lua_session_stop_script_path></*my_lua_session_stop_script_path*>

Session Progress
<lua_session_progress_script_base64></*my_lua_session_progress_script_base64*>
<lua_session_progress_script_path></*my_lua_session_progress_script_path*>

File Start
<lua_file_start_script_base64></*my_lua_file_start_script_base64*>
<lua_file_start_script_path></*my_lua_file_start_script_path*>

File Stop
<lua_file_stop_script_base64></*my_lua_file_stop_script_base64*>
<lua_file_stop_script_path></*my_lua_file_stop_script_path*>

Configuration Examples

Consider, for example, a file called `myluascript`, which contains this code:

```
lua_log("Hello from lua-land")
```

If you wanted to use the base64-encoded version of the script itself, or the encrypted pathname for the script, then the configuration would look something like this:

```
<transfer>
<lua_session_start_script_base64>bHVhX2xvZygiSGVsbG8gZnJvbSBsdWEtbGFuZCIp</
lua_session_start_script_base64>
</transfer>
```

Transfer Session Data Accessible to Scripts

When a Lua script is started, data about the transfer session is automatically written to the (`env_table[]`) table. You can access and modifiy the values in the table with your scripts by using the table index names.

Session, State, and Error Data

env_table[] Index Name	Description	Value
`type`	Event type	`Session` or `Transfer`
`startstop`	Session start or session stop, based on the event `type` value	`Start` or `Stop`
`session_id`	Session ID	*session_d*
`userstr`	A user-defined string; for example, additional variables	*user_string*
`state`	Transfer state	`started` or `success`
`errcode`	Error code	*error_code*
error	Error string associated with the error code	*error_string*

Session Data

env_table[] Index	Description	Value
`transport`	Transport mechanism; either Ascp or Ascp4	`ascp` or `ascp4`
`version`	Ascp or Ascp4 version number	*ascp_or_ascp4_version*
`source`	Source-file pathname	*truncated_source_listing*
`dest`	Destination pathname	*commandline_dest_param*
`local_ip`	Local IP address	*local_IP_address*
`peer_ip`	Peer name or IP address	*peer_name_or_IP_address*
`userid`	User ID	*user_id*
`user`	User name	*user_name*
`direction`	Direction of transfer	`send` or `recv`

env_table[] Index	Description	Value
`cipher`	Encryption cipher for file data, defined with `aspera.conf` or with **ascp** (if both are set, the `aspera.conf` setting overrides **ascp**)	*cipher_name* or `none`
`manifest_file`	The manifest file name, which contains a list of files that have been transferred; only exits if it has been defined with **ascp** `--file-manifest-path-file=`*file_path;* ; the file will have an `.inprogress` extension	*file_path*
`target_rate_kbps`	The maximum target rate for transfers, in Kbps	*target_rate_value_kbps*
`min_rate_kbps`	The initial minumum rate, in Kbps	*initial_minimum_rate_kbps*
`rate_policy`	Defines the **ascp** rate policy. If not defined here, the value is taken from the default configuration in the GUI or `aspera.conf`.	`fixed` or `high` or `regular` or `low`
`filecount`	Number of files	*number_of_files*
`transfer_bytes`	Total number of bytes transferred	*total_bytes_transferred*
`file_bytes`	Total number of bytes written to disk	*total_bytes_written*
`cookie`	The cookie sent to the client system	*user_def_cookie_string*

Transfer Data

env_table[] Index	Description	Value
`direction`	Transfer direction	`send` or `recv`
`file`	Full pathname to file being validated	*file_path*
`size`	File size, in bytes	*file_size_in_bytes*
`startbyte`	Start byte, if resumed	*start_byte_if_resumed, lower range*
`endbyte`	If entire file is not transferred, the last byte	*upper range*
`rate`	Effective rate in Kbps	*effective_rate_in_kbps*
`delay`	Measured network delay	*measured_network_delay*
`loss`	Measured network loss	*measured_network_loss*

env_table[] Index	Description	Value
xfer_id	Transfer UUID	*transfer_uuid*
xfer_retry	The number of retries to attempt, in the event of a failure (**ascp** only)	*retry_number_of_seconds*
tags	Initially set with **ascp**	*transfer_metatags*
file_name_encoding	File encoding	Currently only UTF-8 is used
file_csum	File checksum	*hash_of_file_contents*
file_csum_type	File checksum type	none or *csum_type* or sha256 or sha384 or sha512

Return Values

When a Lua script exits, Lua returns either LRET_OK or LRET_ERROR (followed by an error number or descriptive string.

Ascp and Ascp4 Lua Functions

Ascp and Ascp4 provide Lua functions for filesystem operations, logging, and aborting a session.

Filesystem Functions

The file functions perform operations on files stored in the native file system or Cloud storage:

- `lua_stat()`
- `lua_file_delete()`
- `lua_rename()`

Note: `lua_rename()` is supported for Ascp, but not for Ascp4.

Each of the file functions take a pathname as its only parameter. If docroot is set, the pathname is relative to docroot. For example:

The `lua_stat()` function populates the `stat_data[]` table. The data in the table can be accessed by using the name indices. For example, `stat_data["size"]` holds the file size.

stat_data[] Index	Description	Values
exists	File exists, or not	`true` or `false`
size	File size	*filesize*
blocks	Number of blocks	*fileblocks*
blocksize	Block size	*blocksize*
type	File type	`Invalid` or `S_IFDIR` or `S_IFREG` or `S_IFCHR` or `S_IFBLK` or `S_IFIFO` or `S_IFSOCK` or `S_IFLNK` or `Blockstream` or `Custom` or `Unknown`
modeformat	OS file format	`Windowsformat` or `Linuxformat`
mode	File mode set in filesystem	*filemode* (format based on OS *modeformat*, above)

stat_data[] Index	Description	Values
uid	User ID	*uid*
gid	Group ID	*gid*
ctime	Change time	*ctime*
mtime	Modify time	*mtime*
atime	Access time	*atime*

Logging Functions

The `lua_log()` function writes a string to the various Ascp (or Ascp4) log interfaces. Formatted strings are not supported. Only simple text strings like this can be used:

```
lua_log("Hello from lua-land")
```

This `lua_log()` call creates the following log entry (where `xxxxxx` is a placeholder for time, thread ID, and so on):

```
xxxxxx LOG lua: Hello from lua-land
```

Abort Functions

The `lua_session_abort()` aborts a transfer session. It takes a text string as its parameter, which should be used to define the reason for aborting. For example:

```
lua_session_abort("aborted session: because, because, because!")
```

The `lua_session_set_max_wait()` function sets a countdown time before shutting down. It takes a number for count of seconds; the default is 10 seconds. The `lua_session_set_max_wait()` function is useful when you are working with scripts that run a long time (for example, for validation). The peer is not affected by the countdown, and could shut down (cleanly) while the Lua processing is still underway. Thus runtime information on the peer may not show the results of any Lua call to `lua_session_abort()` should it happen after the session terminates.

Encryption-at-Rest

Lua Function Usage Example

This script first checks for the existence of a file. If the file does not exist, it aborts the transfer session. If the file exists, the script renames it and logs the re-naming activity.

Aspera Watch Service and Watch Folders

Aspera Watch Service and Watch Folders

Watch Folders and the Aspera Watch Service

Introduction to Watch Folders and the Aspera Watch Service

Watch Folders and the Aspera Watch Service offer tools for easily monitoring file system changes in real-time and automatically transferring new and modified files.

Watch Folders

Watch Folders enables large-scale, automated file and directory transfers, including ultra-large directories with over 10 million items and directories with "growing" files. Watch Folders use input from asperawatchd to automate transfers of files added to or modified in a source folder. They can be configured to push from the local server or pull from a remote server. Remote servers can be HSTS, HSTE, and IBM Aspera Shares servers, as well as servers in object storage. Push Watch Folders can use IBM Aspera on Cloud and IBM Aspera Transfer Cluster Manager nodes for a destination.

For more information, see:

- “Watch Folders” on page 186
- IBM Aspera Console Admin Guide: Working with Watch folders

Aspera Watch Service

The Aspera Watch Service (**asperawatchd**) is a file system change detection and snapshot service that is optimized for speed, scale, and distributed sources. On file systems that have file system notifications, changes in source file systems (new files and directories, deleted items, and renames) are detected immediately, eliminating the need to scan the file system. On file systems without file notifications, such as object storage, Solaris, and AIX file system scans are automatically triggered.

The Aspera Watch Service monitors changes to the file system by taking snapshots and analyzing the difference between them. Users create watches by subscribing to a watch service and specifying the part of the file system to watch. You can use the output from asperawatchd to generate a source file list for **ascp** and **async** sessions.

For more information, see:

- “Starting Aspera Watch Services and Creating Watches” on page 233
- “Watch Service Configuration” on page 235
- “Transferring and Deleting Files with the Aspera Watch Service” on page 238
- “Sync with Aspera Watch Service Session Examples” on page 282

Aspera Run Service (`asperarund`)

The asperarund manages both asperawatchd and asperawatchfolderd. It stores both their configurations in its database, automatically starts the services when they are added, and restarts services if they fail. It also enables admins to start services under different users without switching between accounts, and to apply logging and database configurations to all services.

Similar to other Aspera services, asperarund starts automatically upon installation and runs as a system daemon (`asperarund`).

For more information on `asperarund`, see “Creating, Managing, and Configuring Services” on page 231.

Enable Credential Encryption for Watch Folders

In order to use Watch Folders, you must enable encryption for user-credentials (usernames and passwords).

About this task

These are the minimal steps required. The commands must be executed with root permissions. For detailed information, see “Secrets Management with askmscli” on page 117.

Procedure

1. Enable dynamic token generation.

```
# asconfigurator -x "set_node_data;token_dynamic_key,true"
```

2. Restart asperanoded.

```
# systemctl restart asperanoded
```

3. Set up a Redis master key.

```
# export redis_master_key="`/usr/bin/openssl rand -base64 32`"
echo -n $redis_master_key | sudo /opt/aspera/bin/askmscli -s redis-master-key
```

4. Set up key stores for the system users who will use push and pull Watch Folders.

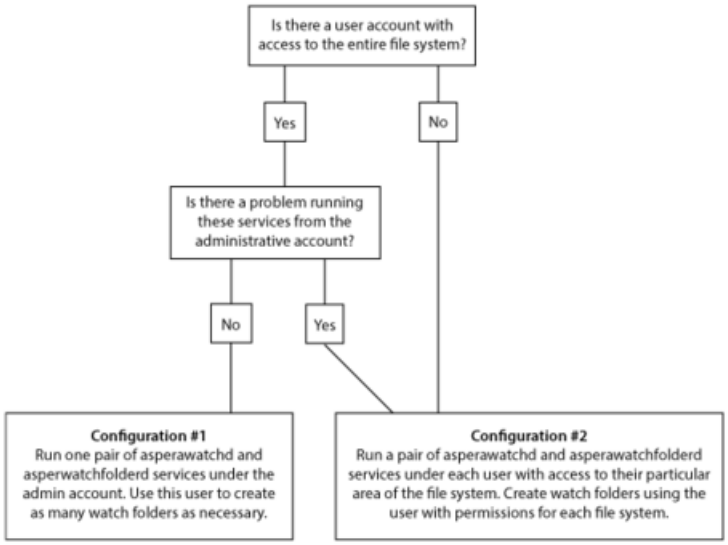
```
#/opt/aspera/bin/askmscli -i -u system_user_name
```

Choosing User Accounts to Run Watch Folder Services

Run asperawatchd and asperawatchfolderd under a user with access to the area of your file system in which you intend to create a watch and Watch Folder. In most cases, the services are run under one user who has access to your entire file system, and watches and Watch Folders are created for specific areas of the file system.

You can also run multiple Watch and Watch Folder services under different users if that is required by your storage configuration or user access restrictions. For example, if your file system includes different mounted storages and no single user can access files in all mounted storages, or if your administrative account has access to the entire file system but your policy prohibits running the services under that user account.

Configure services depending on your user account scenario:



Configuration #1

This is the simplest and most common configuration of Watch Folder services. Use an account that has read permissions for all your files and follow the instructions in “Creating a Push Watch Folder with aswatchfolderadmin” on page 187.

Configuration #2

If you cannot run Watch Folder services under the administrative account or you do not have a single user that has access to the entire file system, run pairs of asperawatchd and asperawatchfolderd under enough users to access your entire file system.

For example, if you have mounted storage from the marketing department that can only be accessed by user xasp1, and another storage from the release team, which can only be accessed by user xasp2, run a pair of asperawatchd and asperawatchfolderd under each user. Aspera recommends using the Node API to configure services and manage Watch Folders in a multi-user context. You can interact with the Node API by using IBM Aspera Console or using **`curl`** commands from the command line.

For more information on using Watch Folders with Console, see "Working with Watch Folders" in the IBM Aspera Console Admin Guide.

Configuring Transfer User Accounts for Watch Folders
In order to use watch folders, you must create or identify a transfer user, and set docroot or file restriction for them.

About this task

Procedure

1. Select or create a user account to run your services.

   Watch Folder services must be run under a user with access to every area of your file system in which you intend to create a Watch Folder. You can run multiple instances of these services under different users; however, most deployments run these services under one user. Choose a user that has access to your entire file system. For information about creating users, see “Setting Up Users” on page 20.

   If you need to run multiple instances of these services to access every area of your file system, see “Choosing User Accounts to Run Watch Folder Services” on page 184.
2. Configure a docroot or file restriction for the user.

   Docroots and path restrictions limit the area of a file system or object storage to which the user has access. Users can create Watch Folders and Watch services on files or objects only within their docroot or restriction.

   Note: Users can have a docroot or restriction, but not both or Watch Folder creation fails.

   To set up a docroot from the command line, run the following command:

   ```
   # asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
   ```

   Restrictions must be set from the command line:

   ```
   # asconfigurator -x "set_user_data;user_name,username;file_restriction,|path"
   ```

   The restriction path format depends on the type of storage. In the following examples, the restriction allows access to the entire storage; specify a bucket or path to limit access.

Storage Type	Format Example
local storage	For Unix-like OS: • specific folder: `file:////folder/*` • drive root: `file:////*` For Windows OS: • specific folder: `file:///c%3A/folder/*` • drive root: `file:///c*`

Storage Type	Format Example
Amazon S3 and IBM Cloud Object Storage - S3	`s3://*`
Azure	`azu://*`
Azure Files	`azure-files://*`
Azure Data Lake Storage	`adl://*`
Alibaba Cloud	`oss://*`
Google Cloud	`gs://*`
HDFS	`hdfs://*`

With a docroot or restriction set up, the user is now an Aspera transfer user. Restart asperanoded to activate your change:

Run the following commands to restart `asperanoded`:

```
# systemctl restart asperanoded
```

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

```
# service asperanoded restart
```

Watch Folders

Watch Folders can be created and managed in the command line, using the `aswatchfolderadmin` tool or the Watch Folder API.

Getting Started with Watch Folders

To use Watch Folders from the command line, make sure that asperarund is running, ensure that the user has the appropriate file permissions for the log directory, and optionally configure the Watch Folder services.

Before you begin

Before you can work with Watch Folders, you must configure a transfer user account. See “Configuring Transfer User Accounts for Watch Folders” on page 185.

Procedure

1. Ensure that asperarund is running.

   See “Creating, Managing, and Configuring Services” on page 231.
2. Ensure the user has permissions to write to the default log directory if no directory is specified.

   For more information about configuring log directories, see“Watch Service Configuration” on page 235.
3. Optionally, configure asperawatchd and asperawatchfolderd settings.

   The default values are already optimized for most users, but you can also configure the snapshot database, snapshot frequency, logging, scan threads, and drop handling, among other features. For instructions, see “Watch Service Configuration” on page 235 and “Watch Folder Service Configuration” on page 195.

Results

Your system is now ready for Watch Folders.

To create a push Watch Folder, see “Creating a Push Watch Folder with aswatchfolderadmin” on page 187 or “Creating a Push Watch Folder with the API” on page 215.

To create a pull Watch Folder, see “Creating a Pull Watch Folder with aswatchfolderadmin” on page 191 or “Creating a Pull Watch Folder with the API” on page 219.

Creating a Push Watch Folder with aswatchfolderadmin

These instructions describe how to create a push Watch Folder by using the **aswatchfolderadmin** utility. **aswatchfolderadmin** requires a JSON configuration file with the syntax introduced in 3.8.0 (described in the following section). Push Watch Folders can still be created from JSON configuration files that follow the 3.7 version syntax by using the Watch Folder API.

About this task

To create and manage Watch Folders by using the Watch Folder API or IBM Aspera Console, see “Creating a Push Watch Folder with the API” on page 215 and the IBM Aspera Console Admin Guide.

When you create a Watch Folder, a Watch service subscription is automatically created to monitor the source directory. In the rare case that the subscription is somehow deleted or impaired, Watch Folders automatically creates a new subscription; however, the new subscription does not retain the file change history and all files in the source directory are re-transferred.

Restrictions on all Watch Folders

- Only local-to-remote (push) and remote-to-local (pull) configurations are supported. Remote-to-remote and local-to-local are not supported.
- Growing files are only supported for local sources (push Watch Folders) and must be authenticated by a transfer user (password or SSH key file). The transfer user cannot be restricted to aspshell and the source cannot be in object storage.
- Source file archiving is not supported if the Watch Folder source is in object storage.
- IBM Aspera Shares endpoints must have version Shares version 1.9.11 with the Watch Folder patch or a later version.

To create a push Watch Folder:

Procedure

1. Prepare your computer as described in “Getting Started with Watch Folders ” on page 186.
2. Create a Watch Service and Watch Folder service for your user on the local computer.

```
# /opt/aspera/sbin/asperawatchd --user username
# /opt/aspera/sbin/asperawatchfolderd --user username
```

3. Verify that the services are running for the user.

```
# /opt/aspera/bin/asrun send -l
```

The output is similar to the following:

```
[asrun send] code=0
{
    "services": [
         {
              "id":"52ca847a-6981-47e1-9f9b-b661cf298af1",
              "configuration": {
                     "enabled":true,
                     "run_as": {
                          "pass": "*****",
                          "user":"root"
                      },
                      "type":"WATCHD"
               },
               "state":"RUNNING",
               "state_changed_at":"2016-10-20T19:14:34Z"
          },
         {
               "id":"d109d1bd-7db7-409f-bb16-ca6ff9abb5f4",
               "configuration": {
                     "enabled":true,
```

```
                    "run_as":{
                         "pass": "*****",
                         "user":"root"
                     },
                     "type":"WATCHFOLDERD"
                },
                "state":"RUNNING",
                "state_changed_at":"2016-10-20T00:11:19Z"
            }
     ]
}
```

Use the **aswatchadmin** and **aswatchfolderadmin** utilities to retrieve a list of running daemons. Daemons have the same name as the user for which they are running. For example, if you used the root user to run your services, you should see the root daemon listed when you run the following commands:

```
# /opt/aspera/bin/aswatchadmin query-daemons
[aswatchadmin query-daemons] Found a single daemon:
    root

# /opt/aspera/bin/aswatchfolderadmin query-daemons
[aswatchfolderadmin query-daemons] Found a single daemon:
    root
```

4. Create a JSON configuration file for your Watch Folder.

   The Watch Folder JSON file describes the source, target, and authentication to the remote server, and can also specify transfer session settings, file handling and post-processing, filters, and growing file handling.

   A basic push Watch Folder configuration file has the following syntax:

```
{
    "source": {
        "path": "source_directory"
    },
    "target": {
        "path": "target_directory",
        "location": {
            "type": "REMOTE",
            "host": "hostname",
            "port": port,
            "authentication": {
                "type": "authentication_mode",
                "user": "username",
                "pass": "password"
                "keypath": "key_file"
              }
        }
    },
    "watchd": {
        "scan_period": "scan_period"
    }
}
```

   For a full configuration reference, see “Watch Folder JSON Configuration File Reference” on page 196.

Field	Description	Default
source path	The local source directory. If the transfer user who is associated with the Node API user is configured with a docroot, then the path is relative to that docroot. If the transfer user is configured with a restriction, then the path is the absolute or UNC path.	N/A
target path	The remote target directory. For SSH and Node API user authentication, the path is relative to the user's docroot, or the absolute path if the transfer user is configured with a restriction. For Shares authentication, the path is the share name and, optionally, a path within the share. For access key authentication, the path is relative to the storage specified in the access key.	N/A

Field	Description	Default
location type	Set "type" to "REMOTE" for the remote server. "type": "REMOTE" is assumed if "host" is specified.	"REMOTE"
host	The host IP address, DNS, hostname, or URL of the remote file system. **Required**. The host can be specified with an IPv4 or IPv6 address. The preferred format for IPv6 addresses is x:x:x:x:x:x:x:x, where each of the eight x is a hexadecimal number of up to 4 hex digits. Zone IDs (for example, %eth0) can be appended to the IPv6 address.	N/A
port	The port to use for authentication to the remote file system. By default, if the authentication type is SSH, then the SSH port for the **ascp** process (the value for tcp_port in the "transport" section) is used. If the authentication type is NODE_BASIC, 9092 is used. For Shares, IBM Aspera Transfer Cluster Manager, or IBM Aspera on Cloud endpoints, enter 443.	If authentication type is SSH, then default is the value for tcp_port in the "transport" section (default: 22). If authentication type is NODE_BASIC, then default is 9092.
authentication type	How Watch Folders authenticates to the remote server. Valid values are SSH or NODE_BASIC. For SSH, authenticate with a transfer user's username and password, or specify the username and the path to their SSH private key file. For NODE_BASIC, authenticate with a Node API username and password, Shares credentials, or an access key ID and secret. Sample JSON syntax for each authentication type is provided following this table.	NODE_BASIC
user	The username for authentication. **Required**. Depending on the type of authentication, it is the transfer user's username, Node API username, Shares username, or access key ID.	N/A
pass	The password for authentication. Depending on the type of authentication, it is the transfer user's password, the Node API user's password, the Shares user's password, or the access key secret. **Required** for SSH authentication if "keypath" is not specified	N/A
keypath	For SSH authentication with an SSH key, the path to the transfer user's SSH private key file. **Required** for SSH authentication if "pass" is not specified	N/A
watchd identifier	The daemon associated with the Watch Service that is used to monitor the file system. Optional. Required only when you want to use a Watch Service that is run by a user who is not associated with the Node API user or access key. Use to specify the daemon on the remote host if it is not **xfer**.	N/A

Field	Description	Default
scan_period	The time between file system scans of the watches (from end of one to start of the next). These scans are independent of the snapshot minimum interval and snapshot minimum changes to ensure that changes are identified. To never scan (asperawatchd relies entirely on file notifications), set to "infinite". On file systems without file notifications, such as object storage, mounted storage (NFS), Solaris, and AIX file system scans triggered by the scan period are used to detect file changes. In this case, set the scan period to frequently scan for changes. On operating systems that support file notifications (Linux, Windows, macOS), asperawatchd uses the file notifications as the primary means for detecting changes, and the scan period serves as a backup. In this case, the default value of 30 minutes is usually acceptable and no change is necessary. To never scan, and rely entirely on file notifications, set to `infinite`. For pull Watch Folders, file systems scans that are triggered by scan_period are the sole means for detecting changes in the source directory. Lower scan periods detect changes faster but can result in greater resource consumption, particularly for object storage. **Note:** The value for scan period cannot be empty, otherwise the configuration is rejected.	30m

Save the configuration file. The path to the configuration file is used in the next step.

5. Create the Watch Folder.

```
# /opt/aspera/bin/aswatchfolderadmin create-folder daemon -f json_file
```

Where *daemon* is the user that is running the Watch Folder services and *json_file* is the path to the Watch Folder configuration file. If you do not know the daemon, retrieve a list of running daemons by running the following command:

```
# /opt/aspera/bin/aswatchfolderadmin query-daemons
[aswatchfolderadmin query-daemons] Found a single daemon:
    root
```

Daemons have the same name as the user for which they are running. For example, if you used the root user to run your services, you should see the root daemon listed.

For example, using the root daemon and a valid JSON file, `watchfolderconf.json`, the output of the **aswatchfolderadmin** command should look like the following:

```
# /opt/aspera/bin/aswatchfolderadmin create-folder root -f watchfolder_conf.json
                                [aswatchfolderadmin create-folder] Successfully created instance
b394d0ee-1cda-4f0d-b785-efdc6496c585.
```

If **aswatchfolderadmin** returns `err=28672`, confirm that the user's docroot allows access to the source directory. If you need to make changes to your docroot, see “Updating the Docroot or Restriction of a Running Watch Folder Service” on page 229.

If **aswatchfolderadmin** returns `err=2`, a Watch Service is not running for the user. See the previous section for instructions on starting a Watch Service.

6. Verify that the Watch Folder is running.

To retrieve a list of running Watch Folders, run the following command:

```
# /opt/aspera/bin/aswatchfolderadmin query-folders daemon_name
```

For example:

```
# /opt/aspera/bin/aswatchfolderadmin query-folders root
                              [aswatchfolderadmin query-folders] Found a single watchfolder:
                              b394d0ee-1cda-4f0d-b785-efdc6496c585
```

7. Test your Watch Folder.

   If the source directory is empty, add files to it. If the configuration is correct, Watch Folders detects the new files, starts a transfer, and the new files appear in the target directory.

   If the source directory is not empty, open the target directory to view files that are automatically transferred as Watch Folders starts.

Results

Once Watch Folders are created, manage them by using the **aswatchfolderadmin** utility. For information, see "Managing Watch Folders with aswatchfolderadmin" on page 213.

Creating a Pull Watch Folder with aswatchfolderadmin

These instructions describe how to create a pull Watch Folder by using the **aswatchfolderadmin** utility. **aswatchfolderadmin** requires a JSON configuration file with the syntax introduced in 3.8.0 (described in the following section). Pull Watch Folders can still be created from JSON configuration files that follow the 3.7 version syntax by using the Watch Folder API.

About this task

To create and manage Watch Folders by using the Watch Folder API or IBM Aspera Console, see "Creating a Push Watch Folder with the API" on page 215 and the IBM Aspera Console Admin Guide.

When you create a Watch Folder, a Watch service subscription is automatically created to monitor the source directory. In the rare case that the subscription is somehow deleted or impaired, Watch Folders automatically creates a new subscription; however, the new subscription does not retain the file change history and all files in the source directory are re-transferred.

Restrictions on all Watch Folders

- Only local-to-remote (push) and remote-to-local (pull) configurations are supported. Remote-to-remote and local-to-local are not supported.
- Growing files are only supported for local sources (push Watch Folders) and must be authenticated by a transfer user (password or SSH key file). The transfer user cannot be restricted to aspshell and the source cannot be in object storage.
- Source file archiving is not supported if the Watch Folder source is in object storage.
- IBM Aspera Shares endpoints must have version Shares version 1.9.11 with the Watch Folder patch or a later version.

Restrictions on Pull Watch Folders

- The remote server must be running HSTS or HSTE version 3.8.0 or newer.
- Pull Watch Folders must be authenticated with an access key ID and secret, a Node API username and password, or IBM Aspera Shares credentials. SSH authentication is not supported for remote sources.
- Pull Watch Folders that use Node API authentication cannot be authenticated with a Node API user whose associated transfer user is configured with a restriction (the Watch Folder status is reported as impaired). Edit the transfer user's configuration to use a docroot, restart asperanoded, and the Watch Folder recovers automatically.
- Pull Watch Folders cannot use IBM Aspera on Cloud (including IBM Aspera on Cloud transfer service nodes) or IBM Aspera Transfer Cluster Manager nodes as the remote source.
- Pull Watch Folders do not support growing files.

To create a pull Watch Folder:

Procedure

1. Create a Watch Service on the remote server.

   If you have SSH access to the server, create the service from the server's command line.

   a) Create the service.

   ```
   # /opt/aspera/sbin/asperawatchd --user username
   ```

   The *username* is for a system user with permissions to the source path.

   b) Confirm that the service was created.

   ```
   # /opt/aspera/bin/aswatchadmin query-daemons
   ```

   If the service exists, the following output is returned (in this example, the user is "root"):

   ```
   # /opt/aspera/bin/aswatchadmin query-daemons
   [aswatchadmin query-daemons] Found a single daemon:
        root
   ```

   If other services are running on the server, other daemons are also returned.

   If you do not have SSH access to the server, use the Node API from your local computer to create the service. This approach requires that you have node credentials for the server. For instructions, see “Creating a Pull Watch Folder with the API” on page 219.

2. Create a Watch Folder service for your user on the local computer.

   ```
   # /opt/aspera/sbin/asperawatchfolderd --user username
   ```

3. Verify that the service is running for the user.

   ```
   # /opt/aspera/bin/asrun send -l
   ```

   The output is similar to the following (in this example, the user is "root"):

   ```
   [asrun send] code=0
   {
       "services": [
            {
                  "id":"d109d1bd-7db7-409f-bb16-ca6ff9abb5f4",
                  "configuration": {
                        "enabled":true,
                        "run_as":{
                             "pass": "*****",
                             "user":"root"
                         },
                         "type":"WATCHFOLDERD"
                  },
                  "state":"RUNNING",
                  "state_changed_at":"2016-10-20T00:11:19Z"
             }
        ]
   }
   ```

   Use the **aswatchadmin** and **aswatchfolderadmin** utilities to retrieve a list of running daemons. Daemons have the same name as the user for which they are running. For example, if you used the root user to run your services, you should see the root daemon listed when you run the following commands:

   ```
   # /opt/aspera/bin/aswatchadmin query-daemons
   [aswatchadmin query-daemons] Found a single daemon:
       root

   # /opt/aspera/bin/aswatchfolderadmin query-daemons
   [aswatchfolderadmin query-daemons] Found a single daemon:
       root
   ```

4. Create a JSON configuration file for your Watch Folder.

The Watch Folder JSON file describes the source, target, and authentication to the remote server, and can also specify transfer session settings, file handling and post-processing, filters, and growing file handling.

A basic pull Watch Folder configuration has the following syntax:

```
{
    "source": {
        "path": "source_directory",
        "location": {
            "type": "REMOTE",
            "host": "ip_address",
            "port": port,
            "authentication": {
                "type": "authentication_mode",
                "user": "username",
                "pass": "password"
             }
        }
    },
    "target": {
        "path": "target_directory"
    },
    "watchd": {
        "scan_period": "scan_period",
        "identifier": "daemon"
    }
}
```

For a full configuration reference, see “Watch Folder JSON Configuration File Reference” on page 196.

Field	Description	Default
source path	The source directory on the remote server. For SSH and Node API user authentication, the path is relative to the associated transfer user's docroot, or the absolute path if the transfer user is configured with a restriction. For Shares authentication, the path is the share name and, optionally, a path within the share. For access key authentication, the path is relative to the storage specified in the access key.	N/A
location type	Set `"type"` to `"REMOTE"` for the remote server. `"type": "REMOTE"` is assumed if `"host"` is specified.	`"REMOTE"`
host	The host IP address, DNS, hostname, or URL of the remote file system. **Required**. The host can be specified with an IPv4 or IPv6 address. The preferred format for IPv6 addresses is x:x:x:x:x:x:x:x, where each of the eight x is a hexadecimal number of up to 4 hex digits. Zone IDs (for example, %eth0) can be appended to the IPv6 address.	N/A
port	The port to use for authentication to the remote file system. By default, if the authentication type is SSH, then the SSH port for the **ascp** process (the value for `tcp_port` in the "transport" section) is used. If the authentication type is `NODE_BASIC`, 9092 is used. For Shares, IBM Aspera Transfer Cluster Manager, or IBM Aspera on Cloud endpoints, enter 443.	If authentication type is SSH, then default is the value for `tcp_port` in the "transport" section (default: 22). If authentication type is `NODE_BASIC`, then default is 9092.
authentication type	How Watch Folders authenticates to the remote server. Pull Watch Folders must use `NODE_BASIC` and authenticate	`NODE_BASIC`

Field	Description	Default
	with a Node API username and password, Shares credentials, or an access key ID and secret.	
user	The username for authentication. **Required**. Depending on the type of authentication, it is the transfer user's username, Node API username, Shares username, or access key ID.	N/A
pass	The password for authentication, depending on the type of authentication.	N/A
target path	The target directory on the local computer, relative to the transfer user's docroot.	N/A
watchd identifier	The daemon associated with the Watch Service that is used to monitor the file system. Optional. Required only when you want to use a Watch Service that is run by a user who is not associated with the Node API user or access key.	The system user that is associated with the Node API user or access key.
scan_period	The time between file system scans of the watches (from end of one to start of the next). These scans are independent of the snapshot minimum interval and snapshot minimum changes to ensure that changes are identified. To never scan (asperawatchd relies entirely on file notifications), set to "infinite". On file systems without file notifications, such as object storage, mounted storage (NFS), Solaris, and AIX file system scans triggered by the scan period are used to detect file changes. In this case, set the scan period to frequently scan for changes. On operating systems that support file notifications (Linux, Windows, macOS), asperawatchd uses the file notifications as the primary means for detecting changes, and the scan period serves as a backup. In this case, the default value of 30 minutes is usually acceptable and no change is necessary. To never scan, and rely entirely on file notifications, set to `infinite`. For pull Watch Folders, file systems scans that are triggered by scan_period are the sole means for detecting changes in the source directory. Lower scan periods detect changes faster but can result in greater resource consumption, particularly for object storage. **Note:** The value for scan period cannot be empty, otherwise the configuration is rejected.	30m

Save the configuration file. The path to the configuration file is used in the next step.

5. Create the Watch Folder.

```
# /opt/aspera/bin/aswatchfolderadmin create-folder daemon -f json_file
```

Where *daemon* is the user that is running the Watch Folder services and *json_file* is the path to the Watch Folder configuration file. If you do not know the daemon, retrieve a list of running daemons by running the following command:

```
# /opt/aspera/bin/aswatchfolderadmin query-daemons
[aswatchfolderadmin query-daemons] Found a single daemon:
    root
```

Daemons have the same name as the user for which they are running. For example, if you used the root user to run your services, you should see the root daemon listed.

For example, using the root daemon and a valid JSON file, `watchfolderconf.json`, the output of the **aswatchfolderadmin** command should look like the following:

```
# /opt/aspera/bin/aswatchfolderadmin create-folder root -f watchfolder_conf.json
                              [aswatchfolderadmin create-folder] Successfully created instance
b394d0ee-1cda-4f0d-b785-efdc6496c585.
```

If **aswatchfolderadmin** returns `err=28672`, confirm that the user's docroot allows access to the source directory. If you need to make changes to your docroot, see "Updating the Docroot or Restriction of a Running Watch Folder Service" on page 229.

If **aswatchfolderadmin** returns `err=2`, a Watch Service is not running for the user. See the previous section for instructions on starting a Watch Service.

6. Verify that the Watch Folder is running.

   To retrieve a list of running Watch Folders, run the following command:

   ```
   # /opt/aspera/bin/aswatchfolderadmin query-folders daemon_name
   ```

   For example:

   ```
   # /opt/aspera/bin/aswatchfolderadmin query-folders root
                                 [aswatchfolderadmin query-folders] Found a single watchfolder:
                                 b394d0ee-1cda-4f0d-b785-efdc6496c585
   ```

7. Test your Watch Folder.

   If the source directory contains files, the Watch Folder collects them into a drop after the Watch service scan interval passes and transfers them to the target.

   Note: No files are transferred until the first scan interval passes. If the Watch service scan interval is set to the default, files transfer after 30 minutes.

Results

Once Watch Folders are created, manage them by using the **aswatchfolderadmin** utility. For information, see "Managing Watch Folders with aswatchfolderadmin" on page 213.

Watch Folder Service Configuration

The configuration for asperawatchfolderd is in the `<server>` section of `aspera.conf`. It includes drop and file management and enabling the use of raw options (**ascp** options that are not yet directly included in Watch Folders).

```
<server>
    <rund>...</rund>
    <watch>
        <log_level>log</log_level>
        <log_directory>AS_NULL</log_directory>
        <db_spec>redis:host:31415:domain</db_spec>
        <watchd>...</watchd>
        <watchfolderd>
            <remote_tmpdir_conf>hide</remote_tmpdir_conf>
            <purge_drops_max_age>1d</purge_drops_max_age>
            <purge_drops_max_files>9223372036854775807</purge_drops_max_files>
            <raw_options>disable</raw_options>
        </watchfolderd>
```

To view the current settings, run the following command and look for settings that start with `watch` and `watchfolderd`:

```
# /opt/aspera/bin/asuserdata -a
```

Configuring asperawatchfolderd Settings

Configure asperawatchfolderd by using **`asconfigurator`** commands with this general syntax:

```
# /opt/aspera/bin/asconfigurator -x "set_server_data;option,value"
```

Options and values are described in the following table.

Watch Folder Configuration Options

Note: The logging and database configuration settings apply to both asperawatchd and asperawatchfolderd, and are described in "Watch Service Configuration" on page 235.

asconfigurator option **aspera.conf setting**	**Description**	**Default**
watchfolderd_purge_drops_max_age <purge_drops_max_age>	The maximum age of stored drops. Drops older than this age are purged.	1d
watchfolderd_purge_drops_max_files <purge_drops_max_files>	The maximum number files across all drops. When this number is exceeded, drops are purged until the file count is less than the specified number.	9223372036854775807
watchfolderd_raw_options <raw_options>	Enable the use of new **`ascp`** options in Watch Folders-initiated transfers before the options are built into the application. Valid values are `disable` or `enable`.	`disable`

Watch Folder JSON Configuration File Reference

Watch Folders are configured by using a JSON configuration file. This article describes all the available configuration options. For simple push and pull configuration examples that contain only the required options, see "Creating a Push Watch Folder with aswatchfolderadmin" on page 187 and "Creating a Pull Watch Folder with aswatchfolderadmin" on page 191.

To get a complete JSON schema that provides the default values, value options, and a description of each parameter, run the following command:

```
# curl -i -u nodeuser:nodepassword https://{domain}:9092/schemas/watchfolders/configuration
```

Sample JSON Configuration File (Pull Watch Folder with Node Authentication)

```
{
  "source": {
    "path": "/projectA",
    "location": {
      "type": "REMOTE",
      "host": "10.0.111.124",
      "port": 9092,
      "authentication": {
        "type": "NODE_BASIC",
        "user": "nodeuser1",
        "pass": "watchfoldersaregreat",
```

```
      }
    }
  },
  "target": {
    "path": "/projectA"
  },
  "id":"b394d0ee-1cda-4f0d-b785-efdc6496c585",
  "cool_off":"30s",
  "snapshot_creation_period":"10s",
  "meta":{
     "version":0,
     "name":"aspera_watchfolder"
  },
  "drop":{
     "detection_strategy":"COOL_OFF_ONLY",
     "cool_off":"5s"
  },
  "post_processing":{
     "source":{
        "type":"TRANSFER_NONE",
        "archive_dir":"/watchfolder_sessions/{$UUID}_{$DATETIME}"
     }
  },
  "filters":[
     {
        "type":"GLOB",
        "pattern":"*.txt",
        "rule":"INCLUDE"
     },
     {
        "type":"GLOB",
        "pattern":"/**",
        "rule":"EXCLUDE"
     }
  ],
  "packages":{
     "timeout":"10s",
     "parsers":[
        {
           "final_transfer":"LIST",
           "filters":[
              {
                 "type":"REGEX",
                 "rule":"INCLUDE",
                 "pattern":".*\\.txt"
              },
              {
                 "type":"REGEX",
                 "rule":"EXCLUDE",
                 "pattern":".*"
              }
           ]
        }
     ]
  },
  "transport":{
     "host":"",
     "user":"aspx2",
     "pass":"",
     "proxy":"dnat://aspx1:passwordsarecool@localhost:9001",
     "keypath":"",
     "fingerprint":"",
     "cookie":"",
     "tags":{
     },
     "error_handling":{
        "file":{
           "max_retries":3,
           "retry_timeout":"3s"
        },
        "drop":{
           "retry_period":"1m"
        }
     },
     "regular":{
        "max_parallel":10,
        "connect_timeout":"10s",
        "policy":"FAIR",
        "min_rate":"0B",
        "target_rate":"10M",
        "tcp_port":22,
        "udp_port":33001,
```

```
         "read_blk_size":"",
         "write_blk_size":"",
         "datagram_size":"",
         "rexmsg_size":"",
         "cipher":"AES128",
         "overwrite":"DIFF",
         "resume":"NONE",
         "preserve_uid":false,
         "preserve_gid":false,
         "preserve_time":false,
         "preserve_creation_time":false,
         "preserve_modification_time":false,
         "preserve_access_time":false,
         "queue_threshold":"5s",
         "sample_period":"2s"
      },
      "growing_file": {
         "max_parallel":8,
         "policy":"FAIR",
         "min_rate":"",
         "target_rate":"10M",
         "tcp_port":22,
         "udp_port":33001,
         "datagram_size":"",
         "cipher":"AES128",
         "completion_timeout":"5s",
         "memory":"2M",
         "chunk_size":"128K",
         "force_send_after":"2s",
         "filters":[
            {
              "type":"REGEX",
              "rule":"INCLUDE",
              "pattern":".*\\.growing"
            },
            {
              "type":"REGEX",
              "rule":"EXCLUDE",
              "pattern":".*"
            }
         ]
      },
      "watchd" : {
         "scan_period":"30m",
         "identifier" : "root"
   }
}
```

Top Level Configuration

Watch Folders supports transfers between a local server and a remote server. For the local server, Watch Folders requires only the local path, whether it is the source or target. For the remote server, Watch Folders requires the host address, port for authentication, and authentication credentials. In the following example, the source is remote and the target is local.

Note: The header `"X-aspera-WF-version:2017_10_23"` is required when submitting POST, PUT, and GET requests to /v3/watchfolders on servers that are version 3.8.0 or newer. This enables Watch Folders to parse the JSON "source" and "target" objects in the format that was introduced in version 3.8.0.

```
{
   "source": {
      "path": "path",
      "location": {
         "type": "REMOTE",
         "host": "host",
         "port": port,
         "authentication": {
            "type": "SSH|NODE_BASIC",
            "user": "username",
            "pass": "password",
            "keypath": "key_file",
            "fingerprint: "ssh_fingerprint"
         }
      }
   },
   "target": {
      "path": "path"
```

```
    },
    "id":"watchfolder_id",
    "cool_off":"30s",
    "snapshot_creation_period":"10s",
    ...
}
```

Field	Description	Default
path	The source or target directory. **Required**. **Local path:** The path is relative to the docroot of the transfer user associated with the node username. If the transfer user is configured with a restriction, the path is the absolute or UNC path. **Remote path:** For access key authentication, the path is relative to the storage specified in the access key. For SSH and Node API user authentication, the path is relative to the user's docroot, configured, or the absolute or UNC path if the user is configured with a restriction. For IBM Aspera Shares authentication, the path is the share name and, optionally, a path within the share. When asperawatchd detects a new file in the source directory, asperawatchfolderd starts an **ascp** session to transfer the file to target directory. The target directory must be within the docroot or restriction set for the user running asperawatchd.	N/A
location type	Set `"type"` to `"REMOTE"` for the remote server. For push Watch Folders the remote server is the "target", for pull Watch Folders the remote server is the "source". One endpoint must be remote and one must be local. Local-to-local and remote-to-remote Watch Folders are not supported.	`"REMOTE"` is assumed if `"host"` is specified. `"LOCAL"` is assumed if `"REMOTE"` or `"host"` is not specified.
host	The host IP address, DNS, hostname, or URL of the remote file system. **Required**. The host can be specified with an IPv4 or IPv6 address. The preferred format for IPv6 addresses is x:x:x:x:x:x:x:x, where each of the eight x is a hexadecimal number of up to 4 hex digits. Zone IDs (for example, %eth0) can be appended to the IPv6 address.	N/A
port	The port to use for authentication to the remote file system. By default, if the authentication type is SSH, then the SSH port for the **ascp** process (the value for `tcp_port` in the "transport" section) is used. If the authentication type is `NODE_BASIC`, 9092 is used. For Shares, IBM Aspera Transfer Cluster Manager, or IBM Aspera on Cloud endpoints, enter 443.	If authentication type is SSH, then default is the value for `tcp_port` in the "transport" section (default: 22). If authentication type is `NODE_BASIC`, then default is 9092.
authentication type	How Watch Folders authenticates to the remote server. Valid values are SSH or `NODE_BASIC`.	`NODE_BASIC`

Field	Description	Default
	For SSH, authenticate with a transfer user's username and password, or specify the username and the path to their SSH private key file. For `NODE_BASIC`, authenticate with a Node API username and password, Shares credentials, or an access key ID and secret. Sample JSON syntax for each authentication type is provided following this table.	
user	The username for authentication. **Required**. Depending on the type of authentication, it is the transfer user's username, Node API username, Shares username, or access key ID.	N/A
pass	The password for authentication. Depending on the type of authentication, it is the transfer user's password, the Node API user's password, the Shares user's password, or the access key secret. **Required** for SSH authentication if "keypath" is not specified	N/A
keypath	For SSH authentication with an SSH key, the path to the transfer user's SSH private key file. **Required** for SSH authentication if "pass" is not specified	N/A
fingerprint	The SSH fingerprint of the remote server. Aspera strongly recommends using SSH fingerprint for security. If the fingerprint does not match that of the server, the transfer fails with the error "Remote host is not who we expected". For more information, see “Securing Your SSH Server” on page 14 ("Configuring Transfer Server Authentication").	N/A
id	Value used to identify a Watch Folder. If this field is not configured at creation, a UUID is automatically generated for and assigned to the Watch Folder.	N/A
cool_off	How long the Watch Folder service waits for files in the watched folder to stop changing (stabilize) before taking a directory snapshot and creating a drop. Default: 5s.	5s
snapshot_creation_period	The interval during which Watch Folders groups new files in the source directory into a drop. All files in a drop are transferred in the same transfer session, post-processed together, and reported as a unit. Watch Folders uses asperawatchd to detect file system modifications, and continuously creates snapshots to compute the snapshot differential. A small value results in high temporal resolution for detecting file system modifications, whereas a large value improves asperawatchd performance. Default: 3s.	3s

Authentication JSON Syntax

- SSH with password

```
"authentication": {
    "type": "SSH",
    "user": "username",
    "pass": "password",
```

```
    "fingerprint": "server_fingerprint"
}
```

- SSH with SSH key

```
"authentication": {
    "type": "SSH",
    "user": "username",
    "keypath": "key_path",
    "fingerprint": "server_fingerprint"
}
```

- NODE_BASIC with Node API username and password

```
"authentication": {
    "type": "NODE_BASIC",
    "user": "node_username",
    "pass": "node_password",
}
```

- NODE_BASIC with Shares credentials

```
"authentication": {
    "type": "NODE_BASIC",
    "user": "shares_username",
    "pass": "shares_password",
}
```

- NODE_BASIC with access key ID and secret

```
"authentication": {
    "type": "NODE_BASIC",
    "user": "access_key_id",
    "pass": "access_key_secret",
}
```

Meta Fields

```
{
   ...
   "meta":{
      "version":0,
      "name":"aspera_watchfolder"
   },
   ...
}
```

Field	Description	Default
version	Specifies the current version of the configuration. When updating the configuration, this value must match the version stored by the server. Otherwise, the update is rejected.	0
name	The value specified in this field is added to the cookie reported by **ascp**. Optional.	N/A

Drop Fields

Watch Folders groups new or updated files it detects in its source folder into "drops". A drop is defined by the duration set by the `snapshot_creation_period`. All files in a given drop are transferred in the same transfer session, post-processed together, and reported as a unit.

```
{
   ...
   "drop":{
      "detection_strategy":"COOL_OFF_ONLY",
      "cool_off":"5s"
   },
```

```
   ...
}
```

Field	Description	Default
detection_strategy	The strategy that Watch Folders uses to create drops when new files are added to the source folder: • COOL_OFF_ONLY: The drop includes new files added to the source folder within the duration of the cool_off field. • TOP_LEVEL_FILES: Create a drop for each file placed in the top level of the source folder. • TOP_LEVEL_DIRS: Create a drop for each directory added to the top level of the source folder. This drop also includes the sub-directories and files in the top level directory.	COOL_OFF_ONLY
cool_off	The time after the first new file is added to the source file during which any other new files are included in the same drop. This setting is only relevant for the COOL_OFF_ONLY detection strategy. Aspera recommends choosing a multiple of the specified snapshot_creation_period for predictable results.	5s

Post Processing Fields

Optionally, specify post-processing to do after a drop or file is successfully transferred.

```
{
   ...
   "post_processing":{
      "source":{
         "type":"TRANSFER_NONE",
         "archive_dir":"/watchfolder_sessions/{$UUID}_{$DATETIME}"
      }
   },
   ...
}
```

Field	Description	Default
type	The type of post-transfer processing. Files can be archived, deleted, or retained after transfer of a drop. When files are archived or deleted, source sub-directories are also deleted from the source, unless the sub-directories were empty to start. File structure is preserved in the archive. • TRANSFER_NONE: Files stay in the source directory. • TRANSFER_ARCHIVE: Files in the source directory are moved to a final archive after successful transfer. This option is not supported for sources in object storage. • TRANSFER_DELETE: Files in the source directory are deleted after successful transfer once the session completes. • FILE_TRANSFER_DELETE: Files in the source directory are deleted after each successfully transfers, rather than waiting for the session to complete.	TRANSFER_NONE
archive_dir	The destination of archived files, if the archive type is TRANSFER_ARCHIVE. The path can be determined using the following variables:	N/A

Field	Description	Default
	• {$TIMESTAMP} (Drop creation time in seconds since epoch) • {$DAY_OF_MONTH} (Time format for drop's creation time) • {$MONTH} • {$YEAR} • {$HOUR} • {$MINUTE} • {$SECOND} • {$DATETIME} (alias for {$YEAR}{$MONTH}{$DAY_OF_MONTH}-{$HOUR}{$MINUTE}{$SECOND}) • {$UUID} • {$NAME} • {$STATE} • {$FILE::STATE} (such as SUCCEEDED, FAILED)	

Filter Fields

Each filter object must include values for "type", "pattern", and "rule". Filters are applied in order. Watch Folders supports glob and Regex filters. The glob filter system is the same as Ascp; see “Using Filters to Include and Exclude Files” on page 145.

```
{
   ...
   "filters":[
      {
         "type":"GLOB",
         "pattern":"*.txt",
         "rule":"INCLUDE"
      },
      {
         "type":"GLOB",
         "pattern":"/**",
         "rule":"EXCLUDE"
      }
   ],
   ...
}
```

Field	Description	Default
type	The type of filter. Supported filters are GLOB and REGEX.	N/A
pattern	The filter pattern.	N/A
rule	The rule for the filter. Supported rules are INCLUDE and EXCLUDE. **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 /** for glob or .* for Regex.	N/A

Packages Fields

Packages values are used to define an order for the transfer queue. For example, if file B depends on file A, file A must be transferred before File B. Dependencies are defined by package files, where the package

file contains the set of files on which it depends. The package file (by default) is transferred after successfully transferring all the files defined in the package file

```
{
   ...
   "packages":{
      "timeout":"10s",
      "parsers":[
         {
            "final_transfer":"LIST",
            "filters":[
               {
                  "type":"REGEX",
                  "rule":"INCLUDE",
                  "pattern":".*\\.txt"
               },
               {
                  "type":"REGEX",
                  "rule":"EXCLUDE",
                  "pattern":".*"
               }
            ],
            ...
         }
      ]
   },
   ...
}
```

Field	Description	Default
timeout	How long to wait for file dependencies to be satisfied (files that must be transferred before the last file are transferred) before considering the dependency as unsatisfied.	10s
final_transfer	Define the file to transfer last. • LIST: The package file is transferred last, after all files in the package file successfully transfer. • LAST_FILE_IN_LIST: The last file in the package file is transferred last.	LIST
filters	Select files to include in the package as those that match the specified filters. Use the same syntax as in the "filters" object.	N/A

The `transport` object

Use to configure authentication to the remote host.

```
}
   ...
   "transport":{
      "host":"198.51.100.22",
      "user":"aspx2",
      "pass":"XF324cd28",
      "token":"fiewle535etn23TEIW234n5sEWTnseonts",
      "proxy":"dnat://aspx1:XF324cd28@localhost:9001",
      "keypath":"~/.ssh/id_rsa",
      "fingerprint":"stringalsdjkfad",
      "tags":{
           "aspera": {
               "cloud-metadata": [
                    {"location":"tarawera"}
               ]
           }
      },
   ...
   }
}
```

Option	Description	Default
host	The host IP address, DNS, hostname, or URL.	N/A
user	The username for authentication. **Required**. Depending on the type of authentication, it is the transfer user's username, Node API username, Shares username, or access key ID.	N/A
pass	The password for authentication. Depending on the type of authentication, it is the SSH user's password, the Node API user's password, the Shares user's password, or the access key secret. This value is not required for SSH authentication that specifies a value for `"keypath"`.	N/A
token	If required, the token string. Not valid for use with growing files.	N/A
proxy	If using, the address of an IBM Aspera Proxy server. The proxy syntax is: `dnat(s)://`*`user`*`:`*`password`*`@`*`server`*`:`*`port`*	N/A
keypath	If authenticating by SSH user and key, the path to the SSH user's private key file. **Note:** If a relative path is provided, the file at the relative path is checked for existence. If the relative path is not found, `$HOME/.ssh/` is prepended to the relative path.	N/A
fingerprint	The SSH fingerprint of the remote server. Aspera strongly recommends using SSH fingerprint for security. If the fingerprint does not match that of the server, the transfer fails with the error "Remote host is not who we expected". For more information, see “Securing Your SSH Server” on page 14 ("Configuring Transfer Server Authentication").	N/A
tags	Specify custom metadata in JSON format. The tags object is passed directly to the ascp session.	N/A

Error Handling Fields

Watch folder error handling distinguishes between two different error categories:

File-Specific Errors: These errors increase the file retry count every time a failure occurs. When the `max_retries` count is reached, the file is marked as failed and the session attempts to transfer the next file in the drop queue. File-specific error include all errors except the following:

- License error
- Authentication error
- Any other error in establishing an **ascp** session

Other Errors: These errors do not increase the file retry count. If a given error re-occurs again and again, the same file is retried until the drop's `retry_period` is exceeded. Then, the drop is marked as failed.

```
}
   ...
   "transport":{
      ...
      "error_handling":{
         "file":{
            "max_retries":3,
            "retry_timeout":"3s"
         },
         "drop":{
```

```
            "retry_period":"1m"
         }
      },
   ...
   }
}
```

Option	Description	Default
max_retries	How many times to try transferring a file before the file is marked as failed.	3
retry_timeout	How frequently to retry file transfers.	3s
retry_period	If no bytes are transferred during the specified period and no file is completed, the drop and all remaining incomplete files in the drop are marked as failed.	1m

The `regular` object

Use to configure Ascp transfer session options.

```
{
   ...
   "transport":{
      ...
      "regular":{
         "max_parallel":10,
         "connect_timeout":"10s",
         "policy":"FAIR",
         "min_rate":"0B",
         "target_rate":"10M",
         "tcp_port":22,
         "udp_port":33001,
         "read_blk_size":"",
         "write_blk_size":"",
         "datagram_size":"",
         "rexmsg_size":"",
         "cipher":"AES128",
         "overwrite":"DIFF",
         "resume":"NONE",
         "preserve_uid":false,
         "preserve_gid":false,
         "preserve_time":false,
         "preserve_creation_time":false,
         "preserve_modification_time":false,
         "preserve_access_time":false,
         "queue_threshold":"5s",
         "sample_period":"2s",
         "content_protect_password":"ear_password"
         "raw_options":["-L","/tmp/log"],
         "symbolic_links":"follow"
      },
      ...
   }
}
```

Field	Description	Default
max_parallel	The maximum number of concurrent **ascp** sessions that Watch Folders can start.	10
connect_timeout	How long Watch Folders waits before reporting that **ascp** as failed.	10s
policy	Specify how **ascp** manages the bandwidth. The policy can be set to the following values: • FIXED	FAIR

Field	Description	Default
	• `FAIR` • `HIGH` • `LOW`	
min_rate	Attempt to transfer no slower than the specified minimum transfer rate.	0B
target_rate	The target transfer rate. Transfer at rates up to the specified target rate. This option accepts suffixes T for terabits/s, G for gigabits/s, M for megabits/s, K for kilobits/s, or B for bits/s. 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.	10M
tcp_port	The port to use for SSH connections.	22
udp_port	The port to use for UDP connections.	33001
read_blk_size	The read block size.	Default determined by settings in `aspera.conf`
write_blk_size	The write block size.	Default determined by settings in `aspera.conf`
datagram_size	The datagram size (MTU) for FASP.	Uses the detected path MTU.
rexmsg_size	The maximum size of a retransmission request. Maximum: 1440.	Determined by **ascp**
cipher	The encryption cipher that is used to encrypt data in transit. 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 explicitly requires CFB, the transfer fails.	AES128

<table>
<tr><th>Field</th><th>Description</th><th>Default</th></tr>
<tr><td></td><td>

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
<table>
<tr><th>Value</th><th>Description</th><th>Support</th></tr>
<tr><td>AES128
AES192
AES256</td><td>Use the GCM or CFB encryption mode, depending on the server configuration and version (see cipher negotiation matrix).</td><td>All client and server versions.</td></tr>
<tr><td>AES128CFB
AES192CFB
AES256CFB</td><td>Use the CFB encryption mode.</td><td>Clients version 3.9.0 and newer, all server versions.</td></tr>
<tr><td>AES128GCM
AES192GCM
AES256GCM</td><td>Use the GCM encryption mode.</td><td>Clients and servers version 3.9.0 and newer.</td></tr>
<tr><td>NONE</td><td>Do not encrypt data in transit. Aspera strongly recommends against using this setting.</td><td>All client and server versions.</td></tr>
</table>
Client-Server Cipher Negotiation
The following table shows which encryption mode is used depending on the server and client versions and settings:
<table>
<tr><th></th><th>Server, v3.9.0+
AES-XXX-GCM</th><th>Server, v3.9.0+
AES-XXX-CFB</th><th>Server, v3.9.0+
AES-XXX</th><th>Server, v3.8.1 or older
AES-XXX</th></tr>
<tr><td>Client, v3.9.0+
AES-XXX-GCM</td><td>GCM</td><td>server refuses transfer</td><td>GCM</td><td>server refuses transfer</td></tr>
</table>
</td><td></td></tr>
</table>

<table>
<tr><th>Field</th><th>Description</th><th>Default</th></tr>
<tr><td></td><td>
<table>
<tr><th></th><th>Server, v3.9.0+
AES-XXX-GCM</th><th>Server, v3.9.0+
AES-XXX-CFB</th><th>Server, v3.9.0+
AES-XXX</th><th>Server, v3.8.1 or older
AES-XXX</th></tr>
<tr><td>Client, v3.9.0+
AES-XXX-CFB</td><td>server refuses transfer</td><td>CFB</td><td>CFB</td><td>CFB</td></tr>
<tr><td>Client, v3.9.0+
AES-XXX</td><td>GCM</td><td>CFB</td><td>CFB</td><td>CFB</td></tr>
<tr><td>Client, v3.8.1 or older
AES-XXX</td><td>server refuses transfer</td><td>CFB</td><td>CFB</td><td>CFB</td></tr>
</table>
</td><td></td></tr>
<tr><td>overwrite</td><td>Specify whether a file is overwritten if it already exists at the destination. Valid options are:
• NEVER
• ALWAYS
• DIFF
• OLDER
• DIFF+OLDER</td><td>DIFF</td></tr>
<tr><td>resume</td><td>Specify if and how partial transfers are resumed.
• NONE: Always transfer the entire file
• FILE_ATTRIBUTES: Resume if file attributes match.
• SPARSE_CHECKSUM: Resume if file attributes and sparse checksum match.
• FULL_CHECKSUM: Resume if file attributes and full checksum match.</td><td>NONE</td></tr>
<tr><td>preserve_uid</td><td>Preserve the file owner user ID.</td><td>false</td></tr>
<tr><td>preserve_gid</td><td>Preserve the file owner group ID.</td><td>false</td></tr>
<tr><td>preserve_time</td><td>This option is equivalent to configuring preserve_creation_time, preserve_modification_time, and preserve_access_time.</td><td>false</td></tr>
<tr><td>preserve_creation_time</td><td>Set creation time of the destination be set to that of the source. If the destination is a non-Windows host, this option is ignored.</td><td>false</td></tr>
<tr><td>preserve_modification_time</td><td>Set the modification time of the destination file to that of the source.</td><td>false</td></tr>
</table>

Field	Description	Default
preserve_access_time	Set the access time of the destination to that of the source. The destination file has the access time of the source file prior to the transfer.	`false`
queue_threshold	Watch Folders controls the amount of data pushed to **ascp** for transferring. When the capacity is reached, Watch Folders waits before pushing new data. This capacity is based on the effective bandwidth reported by **ascp**.	5s
sample_period	Period used to compute the current bandwidth. Used with `queue_threshold` to compute the amount of data pushed to **ascp**.	2s
content_protect_password	Enter a password to enable client-side encryption at rest. Files that are uploaded to the server are appended with a `.aspera-env` extension. To download and decrypt `.aspera-env` files from the server, the client must provide the password. For more information on client-side encryption at rest, see "Client-Side Encryption-at-Rest (EAR)" on page 156.	N/A
raw_options	Specify **ascp** options and their arguments that are not yet available in Watch Folders to apply to Watch Folder transfers. To use raw options, they must be enabled in the client's `aspera.conf` by running the following command: `# asconfigurator -x "set_central_server_data;raw_options,enable"`	disabled
symbolic_links	Set the symbolic link handling policy, as allowed by the server. Value can be `FOLLOW`, `COPY`, or `SKIP`. On Windows, the only method is `SKIP`. For more information on symbolic link handling, see "Symbolic Link Handling" on page 150.	`FOLLOW`

The `growing` Object

Use to stream growing files from the Watch Folder. If a file does not match the growing file filter, it is transferred by Ascp.

Note: Growing files are only supported for local sources (push Watch Folders) and must be authenticated by a transfer user (password or SSH key file). The transfer user cannot be restricted to aspshell and the source cannot be in object storage.

```
{
   ...
   "transport":{
      ...
      "growing_file":{
         "max_parallel":8,
         "policy":"FAIR",
         "min_rate":"",
         "target_rate":"10M",
         "tcp_port":22,
         "udp_port":33001,
         "datagram_size":"",
         "cipher":"AES128",
         "completion_timeout":"5s",
         "memory":"2M",
         "chunk_size":"128K",
         "force_send_after":"2s",
         "filters":[
            {
              "type":"REGEX",
              "rule":"INCLUDE",
```

```
            "pattern":".*\\.growing"
          },
          {
            "type":"REGEX",
            "rule":"EXCLUDE",
            "pattern":".*"
          }
        ]
      }
    }
}
```

Option	Description	Default
max_parallel	The maximum number of concurrent FASPStream sessions the Watch Folder can initiate.	8
policy	Defines how FASPStream manages the bandwidth. The policy can be set to the following values: • `FIXED` • `FAIR` • `HIGH` • `LOW`	`FAIR`
min_rate	Attempt to transfer no slower than the specified minimum transfer rate.	0B
target_rate	The target transfer rate. Transfer at rates up to the specified target rate. This option accepts suffixes T for terabits/s, G for gigabits/s, M for megabits/s, K for kilobits/s, or B for bits/s. 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.	10M
tcp_port	The port to use for SSH connections.	22
udp_port	The port to use for UDP connections.	33001
datagram_size	The datagram size (MTU) for FASP.	The detected path MTU.
cipher	The encryption cipher that is used to encrypt streamed data in transit, either `NONE` and `AES128`.	AES128
completion_timeout	How long to wait before the session is considered complete. A growing file is considered complete when no new data arrives within the timeout period.	5s
force_send_after	Force FASPStream to send data after the given time, even if the chunk is not full.	2s
memory	The maximum amount of memory FASPStream is allowed to use.	2M
chunk_size	Packet size for transfers over the network.	128K
filters	Select growing files to include in the package as those that match the specified filters. Use the same syntax as in the "filters" object.	N/A

The `watchd` Object

Use to manage watchd services for pull Watch Folders when asperawatchd is run on a different node than asperawatchfolderd.

```
{
    ...
    "watchd": {
        "scan_period": "30m",
        "identifier": "daemon",
        "connection":{
            "type": "NONE|REDIS|NODE",
            "host": "ip_address",
            "port": port,
            "authentication": {
                "type":"NODE_BASIC",
                "user": "node_username",
                "pass": "node_password"
            }
        }
    }
}
```

Option	Description	Default
scan_period	The time between file system scans of the watches (from end of one to start of the next). These scans are independent of the snapshot minimum interval and snapshot minimum changes to ensure that changes are identified. To never scan (asperawatchd relies entirely on file notifications), set to "infinite". On file systems without file notifications, such as object storage, mounted storage (NFS), Solaris, and AIX file system scans triggered by the scan period are used to detect file changes. In this case, set the scan period to frequently scan for changes. On operating systems that support file notifications (Linux, Windows, macOS), asperawatchd uses the file notifications as the primary means for detecting changes, and the scan period serves as a backup. In this case, the default value of 30 minutes is usually acceptable and no change is necessary. To never scan, and rely entirely on file notifications, set to `infinite`. For pull Watch Folders, file systems scans that are triggered by scan_period are the sole means for detecting changes in the source directory. Lower scan periods detect changes faster but can result in greater resource consumption, particularly for object storage. **Note:** The value for scan period cannot be empty, otherwise the configuration is rejected.	30m
identifier	The daemon associated with the Watch Service that is used to monitor the file system. Optional. Required only when you want to use a Watch Service that is run by a user who is not associated with the Node API user or access key.	The system user that is associated with the Node API user or access key.
connection type	The method for connecting to asperawatchd. Value can be `NONE`, `NODE`, or `REDIS`. If `NODE` or `REDIS` is specified, then host and port must also be specified. If `NODE` is specified, then Node API credentials must be specified in the authorization object.	`NONE`

Option	Description	Default
host	The IP address or the URL of the host of asperanoded or the Redis database.	localhost
port	The port for asperanoded or the Redis database. By default, Node uses 9092 and Redis uses 31415.	31415
authentication type	The method for authentication. Only option is `NODE_BASIC`. This value is used only if connection type is `NODE`.	`NODE_BASIC`
user	The Node API username. This value is used only if connection type is `NODE`.	N/A
pass	The Node API password. This value is used only if connection type is `NODE`.	N/A

Managing Watch Folders with aswatchfolderadmin

The **`aswatchfolderadmin`** tool can be used to retrieve a list of Watch Folders, update the configuration of Watch Folder, and delete a Watch Folder.

Retrieve a List of Running Daemons

Use the **`aswatchadmin`** and **`aswatchfolderadmin`** utilities to retrieve a list of running daemons. Daemons have the same name as the user for which they are running. For example, if you used the root user to run your services, you should see the root daemon listed when you run the following commands:

```
# /opt/aspera/bin/aswatchadmin query-daemons
[aswatchadmin query-daemons] Found a single daemon:
    root

# /opt/aspera/bin/aswatchfolderadmin query-daemons
[aswatchfolderadmin query-daemons] Found a single daemon:
    root
```

Retrieve a List of Watch Folders

```
# /opt/aspera/bin/aswatchfolderadmin query-folders daemon
```

For example, if two Watch Folders are configured for the daemon root, the output is similar to the following:

```
# /opt/aspera/bin/aswatchfolderadmin query-folders root
[aswatchfolderadmin query-folders] Found 2 watchfolders:
    3354f360-dfa6-4789-930e-074cd9d4551b
    b394d0ee-1cda-4f0d-b785-efdc6496c585
```

Update a Watch Folder's Configuration

To update a Watch Folder configuration, retrieve the Watch Folder's configuration, make the desired changes, and then save the configuration as a JSON file. You cannot pass a new configuration file to the `update-folder` sub-command, because the new configuration file must match the old file exactly, except for the changes you are making.

1. Retrieve and save the Watch Folder configuration in a new file:

   ```
   # /opt/aspera/bin/aswatchfolderadmin query-folders daemon -i watch_folder_id --config >
   filename.json
   ```

2. Edit the configuration settings in the file.

   Note: When **`aswatchfolderadmin`** returns the JSON configuration, it obfuscates the password for the host with asterisks (******). If you do not want to update the password, leave it obfuscated (as

asterisks) in the new file and the old password is used. To update the password, enter the new string. If no password is specified, then the password value is empty and transfers cannot be authenticated.

3. Save your changes.
4. Submit the updated configuration file to **aswatchfolderadmin**:

```
# /opt/aspera/bin/aswatchfolderadmin update-folder daemon watchfolder_id -f json_file
```

For example:

```
# /opt/aspera/bin/aswatchfolderadmin update-folder root 3354f360-dfa6-4789-930e-074cd9d4551b
-f watchfolder_conf.json
[aswatchfolderadmin update-folder] Successfully updated instance b394d0ee-1cda-4f0d-b785-
efdc6496c585
```

Delete a Watch Folder

```
# /opt/aspera/bin/aswatchfolderadmin delete-folder daemon watchfolder_id
```

For example:

```
# /opt/aspera/bin/aswatchfolderadmin update-folder root 3354f360-dfa6-4789-930e-074cd9d4551b
[aswatchfolderadmin update-folder] Successfully deleted instance b394d0ee-1cda-4f0d-b785-
efdc6496c585
```

Configuring Linux for Many Watch Folders

To run many (>100) push Watch Folders on Linux computers, adjust three system settings and then reload the `sysctl.conf` file to activate them.

Procedure

1. Increase the maximum number of watches allowed by the system.

   Retrieve the current value by running the following command:

```
$ cat /proc/sys/fs/inotify/max_user_watches
8192
```

   To permanently increase the number of available watches (to a value that is greater than the number of files to watch, such as 524288), add the configuration to `/etc/sysctl.conf`:

```
$ sudo echo "fs.inotify.max_user_watches=524288" >> /etc/sysctl.conf
```

2. Increase the maximum number of inotify instances, which correspond to the number of allowed Watch Services instances.

   Retrieve the current value by running the following command:

```
$ cat /proc/sys/fs/inotify/max_user_instances
128
```

   On many systems, the default value is 128, meaning only 128 watches can be created. To permanently increase the number available (to a value that is greater than the number of desired Watch Folder instances, such as 1024), add the configuration to `/etc/sysctl.conf`:

```
$ sudo echo "fs.inotify.max_user_instances=1024" >> /etc/sysctl.conf
```

3. Increase the open file limit.

   Retrieve the current value by running the following command:

```
$ cat /proc/sys/fs/file-max
794120
```

To permanently increase the open file limit (to a value that is greater than the number of desired watches, such as 2097152), add the configuration to `/etc/sysctl.conf`:

```
$ sudo echo "fs.file-max=2097152" >> /etc/sysctl.conf
```

4. Reload systemd settings to activate the new settings.

   To reload systemd settings, either reboot the machine or run the following command:

```
$ sudo sysctl -p /etc/sysctl.conf
```

Creating a Push Watch Folder with the API

These instructions describe how to create a push Watch Folder by using the Watch Folder API.

About this task

You can also create and manage Watch Folders from the command line ("Creating a Push Watch Folder with aswatchfolderadmin" on page 187) or by using IBM Aspera Console (IBM Aspera Console Admin Guide).

When you create a Watch Folder, a Watch service subscription is automatically created to monitor the source directory. In the rare case that the subscription is somehow deleted or impaired, Watch Folders automatically creates a new subscription; however, the new subscription does not retain the file change history and all files in the source directory are re-transferred.

Restrictions on all Watch Folders

- Only local-to-remote (push) and remote-to-local (pull) configurations are supported. Remote-to-remote and local-to-local are not supported.
- Growing files are only supported for local sources (push Watch Folders) and must be authenticated by a transfer user (password or SSH key file). The transfer user cannot be restricted to aspshell and the source cannot be in object storage.
- Source file archiving is not supported if the Watch Folder source is in object storage.
- IBM Aspera Shares endpoints must have version Shares version 1.9.11 with the Watch Folder patch or a later version.

To create a push Watch Folder with the API:

Procedure

1. Prepare your computer as described in "Getting Started with Watch Folders " on page 186.
2. Create a Node API user and associate it with a transfer user account. The user account must have administrative privileges to interact with asperawatchfolderd.

```
# /opt/aspera/bin/asnodeadmin -a -u node_username -p node_password -x admin_user --acl-set
"admin,impersonation"
```

   For example:

```
# /opt/aspera/bin/asnodeadmin -a -u watchfolder_user -p X245lskd3 -x root --acl-set
"admin,impersonation"
```

   Adding, modifying, or deleting a node-user triggers automatic reloading of the user database and the node's configuration and license files. For more information on the Node API, see your transfer server's administrator guide.
3. Verify that you correctly added the Node API user.

```
# /opt/aspera/bin/asnodeadmin -l
                        List of Node API user(s):
                  user       system/transfer user                    acls
====================    ======================    ====================
       node_api_user                 system_user    [admin,impersonation]
```

For example, using the information from the example in the previous step, the output is similar to the following:

```
# /opt/aspera/bin/asnodeadmin -l
                user       system/transfer user                    acls
====================    ======================    ====================
```

4. Create the Watch service and Watch Folder service.
   a) Create a JSON configuration file for each service.

      For the Watch Service:

      ```
      {
          "type": "WATCHD",
          "run_as": {
              "user": "username",
              "pass": "password"
          },
          "enabled": true
      }
      ```

      For the Watch Folder service:

      ```
      {
          "type": "WATCHFOLDERD",
          "run_as": {
              "user": "username",
              "pass": "password"
          },
          "enabled": true
      }
      ```

      The *username* and *password* are for a transfer user with permissions to the source path. Save the files, with the `.json` extension.

   b) To create the services, run the following command for each one:

      ```
      # curl -ki -u node_username:node_password -X POST -d @config_file "https://localhost:9092/
      rund/services"
      ```

      If service creation succeeds, the ID of the service is returned. Record the IDs for use in the next step.

5. Confirm that the services are running.

   For each service, run the following command:

   ```
   # curl -ki -u node_username:node_password -X GET "https://localhost:9092/rund/services/
   service_id"
   ```

   The state is reported as "RUNNING".

6. Create a JSON configuration file for your Watch Folder.

   The Watch Folder JSON file describes the source, target, and authentication to the remote server, and can also specify transfer session settings, file handling and post-processing, filters, and growing file handling.

   A basic push Watch Folder configuration file has the following syntax:

   ```
   {
       "source": {
           "path": "source_directory"
       },
       "target": {
           "path": "target_directory",
           "location": {
               "type": "REMOTE",
               "host": "hostname",
               "port": port,
               "authentication": {
                   "type": "authentication_mode",
                   "user": "username",
   ```

```
                "pass": "password"
                "keypath": "key_file"
              }
        }
    },
    "watchd": {
        "scan_period": "scan_period"
    }
}
```

For a full configuration reference, see “Watch Folder JSON Configuration File Reference” on page 196.

Field	Description	Default
source path	The local source directory. If the transfer user who is associated with the Node API user is configured with a docroot, then the path is relative to that docroot. If the transfer user is configured with a restriction, then the path is the absolute or UNC path.	N/A
target path	The remote target directory. For SSH and Node API user authentication, the path is relative to the user's docroot, or the absolute path if the transfer user is configured with a restriction. For Shares authentication, the path is the share name and, optionally, a path within the share. For access key authentication, the path is relative to the storage specified in the access key.	N/A
location type	Set `"type"` to `"REMOTE"` for the remote server. `"type": "REMOTE"` is assumed if `"host"` is specified.	`"REMOTE"`
host	The host IP address, DNS, hostname, or URL of the remote file system. **Required**. The host can be specified with an IPv4 or IPv6 address. The preferred format for IPv6 addresses is x:x:x:x:x:x:x:x, where each of the eight x is a hexadecimal number of up to 4 hex digits. Zone IDs (for example, %eth0) can be appended to the IPv6 address.	N/A
port	The port to use for authentication to the remote file system. By default, if the authentication type is SSH, then the SSH port for the **ascp** process (the value for `tcp_port` in the "transport" section) is used. If the authentication type is `NODE_BASIC`, 9092 is used. For Shares, IBM Aspera Transfer Cluster Manager, or IBM Aspera on Cloud endpoints, enter 443.	If authentication type is SSH, then default is the value for `tcp_port` in the "transport" section (default: 22). If authentication type is `NODE_BASIC`, then default is 9092.
authentication type	How Watch Folders authenticates to the remote server. Valid values are SSH or `NODE_BASIC`. For SSH, authenticate with a transfer user's username and password, or specify the username and the path to their SSH private key file. For `NODE_BASIC`, authenticate with a Node API username and password, Shares credentials, or an access key ID and secret. Sample JSON syntax for each authentication type is provided following this table.	`NODE_BASIC`

Field	Description	Default
user	The username for authentication. **Required**. Depending on the type of authentication, it is the transfer user's username, Node API username, Shares username, or access key ID.	N/A
pass	The password for authentication. Depending on the type of authentication, it is the transfer user's password, the Node API user's password, the Shares user's password, or the access key secret. **Required** for SSH authentication if "keypath" is not specified	N/A
keypath	For SSH authentication with an SSH key, the path to the transfer user's SSH private key file. **Required** for SSH authentication if "pass" is not specified	N/A
watchd identifier	The daemon associated with the Watch Service that is used to monitor the file system. Optional. Required only when you want to use a Watch Service that is run by a user who is not associated with the Node API user or access key. Use to specify the daemon on the remote host if it is not **xfer**.	N/A
scan_period	The time between file system scans of the watches (from end of one to start of the next). These scans are independent of the snapshot minimum interval and snapshot minimum changes to ensure that changes are identified. To never scan (asperawatchd relies entirely on file notifications), set to "infinite". On file systems without file notifications, such as object storage, mounted storage (NFS), Solaris, and AIX file system scans triggered by the scan period are used to detect file changes. In this case, set the scan period to frequently scan for changes. On operating systems that support file notifications (Linux, Windows, macOS), asperawatchd uses the file notifications as the primary means for detecting changes, and the scan period serves as a backup. In this case, the default value of 30 minutes is usually acceptable and no change is necessary. To never scan, and rely entirely on file notifications, set to `infinite`. For pull Watch Folders, file systems scans that are triggered by scan_period are the sole means for detecting changes in the source directory. Lower scan periods detect changes faster but can result in greater resource consumption, particularly for object storage. **Note:** The value for scan period cannot be empty, otherwise the configuration is rejected.	30m

Save the configuration file. The path to the configuration file is used in the next step.

7. Start the Watch Folder.

```
# curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X POST
-d @path/to/json_file https://host:node_api_port/v3/watchfolders
```

By default, the API port is 9092.

Note: The header `"X-aspera-WF-version:2017_10_23"` is required when submitting POST, PUT, and GET requests to /v3/watchfolders on servers that are version 3.8.0 or newer. This enables Watch Folders to parse the JSON "source" and "target" objects in the format that was introduced in version 3.8.0.

For example:

```
# curl -k --user watchfolder_admin:XF324cd28 -H "X-aspera-WF-version:2017_10_23" -X POST -d
@/watchfolder_conf.json https://198.51.100.22:9092/v3/watchfolders
{
"id": "b394d0ee-1cda-4f0d-b785-efdc6496c585"
}
```

8. Verify that the Watch Folder is running.

```
# curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
https://host:node_api_port/v3/watchfolders/watchfolder_id/state
```

For example:

```
# curl -sk --user watchfolder_admin:XF324cd28  -H "X-aspera-WF-version:2017_10_23" -X GET
https://198.51.100.22:9092/v3/watchfolders/b394d0ee-1cda-4f0d-b785-efdc6496c585/state
```

If the Watch Folder is running, it is reported with `"state":"HEALTHY"`.

Results

You can manage Watch Folders using the API. For more information, see "Managing Watch Folders with the API" on page 223.

Creating a Pull Watch Folder with the API

These instructions describe how to create a pull Watch Folder by using the Watch Folder API.

About this task

You can also create and manage Watch Folders from the command line ("Creating a Pull Watch Folder with aswatchfolderadmin" on page 191) or by using IBM Aspera Console (IBM Aspera Console Admin Guide).

When you create a Watch Folder, a Watch service subscription is automatically created to monitor the source directory. In the rare case that the subscription is somehow deleted or impaired, Watch Folders automatically creates a new subscription; however, the new subscription does not retain the file change history and all files in the source directory are re-transferred.

Restrictions on all Watch Folders

- Only local-to-remote (push) and remote-to-local (pull) configurations are supported. Remote-to-remote and local-to-local are not supported.
- Growing files are only supported for local sources (push Watch Folders) and must be authenticated by a transfer user (password or SSH key file). The transfer user cannot be restricted to aspshell and the source cannot be in object storage.
- Source file archiving is not supported if the Watch Folder source is in object storage.
- IBM Aspera Shares endpoints must have version Shares version 1.9.11 with the Watch Folder patch or a later version.

Restrictions on Pull Watch Folders

- The remote server must be running HSTS or HSTE version 3.8.0 or newer.
- Pull Watch Folders must be authenticated with an access key ID and secret, a Node API username and password, or IBM Aspera Shares credentials. SSH authentication is not supported for remote sources.
- Pull Watch Folders that use Node API authentication cannot be authenticated with a Node API user whose associated transfer user is configured with a restriction (the Watch Folder status is reported as impaired). Edit the transfer user's configuration to use a docroot, restart asperanoded, and the Watch Folder recovers automatically.
- Pull Watch Folders cannot use IBM Aspera on Cloud (including IBM Aspera on Cloud transfer service nodes) or IBM Aspera Transfer Cluster Manager nodes as the remote source.

- Pull Watch Folders do not support growing files.

Procedure

1. Prepare your computer as described in “Getting Started with Watch Folders ” on page 186.
2. Create a Node API user and associate it with a transfer user account. The user account must have administrative privileges to interact with asperawatchfolderd.

   ```
   # /opt/aspera/bin/asnodeadmin -a -u node_username -p node_password -x admin_user --acl-set
   "admin,impersonation"
   ```

   For example:

   ```
   # /opt/aspera/bin/asnodeadmin -a -u watchfolder_user -p X245lskd3 -x root --acl-set
   "admin,impersonation"
   ```

   Adding, modifying, or deleting a node-user triggers automatic reloading of the user database and the node's configuration and license files. For more information on the Node API, see your transfer server's administrator guide.
3. Verify that you correctly added the Node API user.

   ```
   # /opt/aspera/bin/asnodeadmin -l
                          List of Node API user(s):
                  user       system/transfer user                    acls
   ====================    ======================    ====================
          node_api_user                system_user    [admin,impersonation]
   ```

   For example, using the information from the example in the previous step, the output is similar to the following:

   ```
   # /opt/aspera/bin/asnodeadmin -l
                  user       system/transfer user                    acls
   ====================    ======================    ====================
   ```

4. Create a Watch Service on the remote server.

   This approach requires that you have node credentials for the remote server.

   a) Create a JSON configuration file for the remote Watch Service.

   ```
   {
       "type": "WATCHD",
       "run_as": {
           "user": "username",
           "pass": "password"
       },
       "enabled": true
   }
   ```

   The *username* and *password* are for a transfer user with permissions to the source path. Save the file as `wfd_create.json`.

   b) To create the service, run the following command:

   ```
   # curl -ki -u node_username:node_password -X POST -d @wfd_create.json "https://
   server_ip_address:9092/rund/services"
   ```

   The output includes the service ID. Record the ID for the next substep.

   c) Confirm that the service is running.

   ```
   # curl -ki -u node_username:node_password -X GET "https://server_ip_address:9092/rund/
   services/service_id"
   ```

5. Create the Watch Folder service on the local computer.

   a) Create a JSON configuration file for the service with the following text:

```
{
    "type": "WATCHFOLDERD",
    "run_as": {
        "user": "username",
        "pass": "password"
    },
    "enabled": true
}
```

The *username* and *password* are for a transfer user with permissions to the source path. Save the files, with the `.json` extension.

b) Create the service.

```
# curl -ki -u node_username:node_password -X POST -d @config_file "https://localhost:9092/
rund/services"
```

If service creation succeeds, the ID of the service is returned. Record the ID for use in the next step.

c) Confirm that the service is running.

```
# curl -ki -u node_username:node_password -X GET "https://localhost:9092/rund/services/
service_id"
```

6. Create a JSON configuration file for your Watch Folder.

The Watch Folder JSON file describes the source, target, and authentication to the remote server, and can also specify transfer session settings, file handling and post-processing, filters, and growing file handling.

A basic pull Watch Folder configuration has the following syntax:

```
{
    "source": {
        "path": "source_directory",
        "location": {
            "type": "REMOTE",
            "host": "ip_address",
            "port": port,
            "authentication": {
                "type": "authentication_mode",
                "user": "username",
                "pass": "password"
             }
        }
    },
    "target": {
        "path": "target_directory"
    },
    "watchd": {
        "scan_period": "scan_period",
        "identifier": "daemon"
    }
}
```

For a full configuration reference, see “Watch Folder JSON Configuration File Reference” on page 196.

Field	Description	Default
source path	The source directory on the remote server. For SSH and Node API user authentication, the path is relative to the associated transfer user's docroot, or the absolute path if the transfer user is configured with a restriction. For Shares authentication, the path is the share name and, optionally, a path within the share. For access key authentication, the path is relative to the storage specified in the access key.	N/A
location type	Set `"type"` to `"REMOTE"` for the remote server. `"type": "REMOTE"` is assumed if `"host"` is specified.	`"REMOTE"`

Field	Description	Default
host	The host IP address, DNS, hostname, or URL of the remote file system. **Required**. The host can be specified with an IPv4 or IPv6 address. The preferred format for IPv6 addresses is x:x:x:x:x:x:x:x, where each of the eight x is a hexadecimal number of up to 4 hex digits. Zone IDs (for example, %eth0) can be appended to the IPv6 address.	N/A
port	The port to use for authentication to the remote file system. By default, if the authentication type is SSH, then the SSH port for the **ascp** process (the value for `tcp_port` in the "transport" section) is used. If the authentication type is `NODE_BASIC`, 9092 is used. For Shares, IBM Aspera Transfer Cluster Manager, or IBM Aspera on Cloud endpoints, enter 443.	If authentication type is SSH, then default is the value for `tcp_port` in the "transport" section (default: 22). If authentication type is `NODE_BASIC`, then default is 9092.
authentication type	How Watch Folders authenticates to the remote server. Pull Watch Folders must use `NODE_BASIC` and authenticate with a Node API username and password, Shares credentials, or an access key ID and secret.	`NODE_BASIC`
user	The username for authentication. **Required**. Depending on the type of authentication, it is the transfer user's username, Node API username, Shares username, or access key ID.	N/A
pass	The password for authentication, depending on the type of authentication.	N/A
target path	The target directory on the local computer, relative to the transfer user's docroot.	N/A
watchd identifier	The daemon associated with the Watch Service that is used to monitor the file system. Optional. Required only when you want to use a Watch Service that is run by a user who is not associated with the Node API user or access key.	The system user that is associated with the Node API user or access key.
scan_period	The time between file system scans of the watches (from end of one to start of the next). These scans are independent of the snapshot minimum interval and snapshot minimum changes to ensure that changes are identified. To never scan (asperawatchd relies entirely on file notifications), set to "infinite". On file systems without file notifications, such as object storage, mounted storage (NFS), Solaris, and AIX file system scans triggered by the scan period are used to detect file changes. In this case, set the scan period to frequently scan for changes. On operating systems that support file notifications (Linux, Windows, macOS), asperawatchd uses the file notifications as the primary means for detecting changes, and the scan period serves as a backup. In this case, the default value of 30 minutes is usually acceptable and no change is necessary. To never scan, and rely entirely on file notifications, set to `infinite`.	30m

Field	Description	Default
	For pull Watch Folders, file systems scans that are triggered by scan_period are the sole means for detecting changes in the source directory. Lower scan periods detect changes faster but can result in greater resource consumption, particularly for object storage. **Note:** The value for scan period cannot be empty, otherwise the configuration is rejected.	

Save the configuration file. The path to the configuration file is used in the next step.

7. Start the Watch Folder.

```
# curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X POST
-d @path/to/json_file https://host:node_api_port/v3/watchfolders
```

By default, the API port is 9092.

Note: The header "`X-aspera-WF-version:2017_10_23`" is required when submitting POST, PUT, and GET requests to /v3/watchfolders on servers that are version 3.8.0 or newer. This enables Watch Folders to parse the JSON "source" and "target" objects in the format that was introduced in version 3.8.0.

For example:

```
# curl -k --user watchfolder_admin:XF324cd28 -H "X-aspera-WF-version:2017_10_23" -X POST -d
@/watchfolder_conf.json https://198.51.100.22:9092/v3/watchfolders
{
"id": "b394d0ee-1cda-4f0d-b785-efdc6496c585"
}
```

8. Verify that the Watch Folder is running.

```
# curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
https://host:node_api_port/v3/watchfolders/watchfolder_id/state
```

For example:

```
# curl -sk --user watchfolder_admin:XF324cd28  -H "X-aspera-WF-version:2017_10_23" -X GET
https://198.51.100.22:9092/v3/watchfolders/b394d0ee-1cda-4f0d-b785-efdc6496c585/state
```

If the Watch Folder is running, it is reported with "`state":"HEALTHY`".

Results

You can manage Watch Folders using the API. For more information, see “Managing Watch Folders with the API” on page 223.

Managing Watch Folders with the API

You can use the Watch Folder API to create, remove, and manage Watch Folders. The instructions below uses **`curl`** commands to interact with the API.

Retrieve a list of Watch Folders

To retrieve a list of Watch Folders, run the following curl command:

```
# curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
https://host:node_api_port/v3/watchfolders
```

For example:

```
# curl -k --user watchfolder_admin:XF324cd28 -H "X-aspera-WF-version:2017_10_23" -X GET https://
198.51.100.22:9092/v3/watchfolders
{
"ids" : [
"b394d0ee-1cda-4f0d-b785-efdc6496c585"
]
}
```

If there are no running Watch Folders, the server returns the following output.

```
{
"ids" : [

]
}
```

Check state, statistics, and status of a watch, transfer, or Watch Folder

```
curl -ks -u node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET https://
host:node_api_port/v3/watchfolders/watchfolder_id/state
```

In the following example, the output shows Watch Folder errored due to a configuration option that was not set. Errors with **ascp** transfers are displayed similarly in the transport section.

```
# curl -ks --user watchfolder_admin:XF324cd28 -H "X-aspera-WF-version:2017_10_23" -X GET
https://198.51.100.22:9092/v3/watchfolders/b394d0ee-1cda-4f0d-b785-efdc6496c585/state
{
  "state": "HEALTHY",
  "statistics": {
    "files_transferred": 0,
    "files_succeeded": 0,
    "files_failed": 0,
    "files_skipped": 0,
    "files_ignored": 0,
    "files_disappeared_before_cool_off": 0,
    "files_unsatisfied_dependency": 0,
    "files_never_appeared": 0,
    "bytes_completed": 0,
    "bytes_written": 0
  },
  "components": {
    "watch": {
      "state": "HEALTHY",
      "state_changed_at": "2016-12-19T20:18:47Z"
    },
    "transport": {
      "state": "UNKNOWN",
      "state_changed_at": "2016-12-19T20:17:48Z"
    },
    "watchfolderd": {
      "state": "HEALTHY",
      "state_changed_at": "2016-12-19T20:18:47Z",
      "last_error": "UAC don't allow raw_options",
      "last_error_at": "2016-12-19T20:18:10Z"
    }
  }
}
```

Query and save a configuration for a specific Watch Folder

```
# curl -ks -u node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
https://host:node_api_port/v3/watchfolders/watchfolder_id > config_file.json
```

For example:

```
# curl -ks --user watchfolder_admin:XF324cd28 -H "X-aspera-WF-version:2017_10_23" -X GET
https://198.51.100.22:9092/v3/watchfolders/b394d0ee-1cda-4f0d-b785-efdc6496c585 >
wf_config1.json
```

Copy the output in a `.json` file.

Retry a failed drop

Watch Folders groups files into "drops" for transfer. If a file in a drop fails to transfer, it is automatically retried based on the Watch Folder configuration (see options in the "error_handling" section, “Watch Folder JSON Configuration File Reference” on page 196). A drop is marked as failed if the file does not transfer within the specified retry period.

You can retry to transfer the failed drop through the Watch Folder API by retrieving the Watch Folder ID and drop ID, then updating the state of the drop:

1. Get the ID of the Watch Folder that you want to update by getting a list of Watch Folders:

   ```
   # curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
   https://host:node_api_port/v3/watchfolders
   ```

2. Get the ID of the failed drop:

   ```
   # curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
   https://host:node_api_port/v3/watchfolders/watchfolder_id/drops?state="FAILED"
   ```

   If you need to disambiguate failed drops by seeing the files that are contained in them, you can run the following command:

   ```
   # curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
   https://host:node_api_port/v3/watchfolders/watchfolder_id/drops/drop_id/files
   ```

3. Retry the drop by changing the state to RETRY:

   ```
   # curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X PUT
   https://host:node_api_port/v3/watchfolders/watchfolder_id/drops/drop_id -d'{"state":"RETRY"}'
   ```

The drop transfer now retries for the specified number of attempts within the retry period.

Updating a Watch Folder

To update a Watch Folder configuration, retrieve the Watch Folder's configuration, make the desired changes, and then save the configuration as a JSON file.

You cannot use a new configuration file, because the new configuration file must match the old file exactly, except for the changes you are making, and because the configuration version number increments with each update.

1. Get the ID of the Watch Folder that you want to update by getting a list of Watch Folders:

   ```
   # curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
   https://host:node_api_port/v3/watchfolders
   ```

2. Save the Watch Folder configuration file for editing:

   ```
   # curl -ks -u node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X GET
   https://host:node_api_port/v3/watchfolders/watchfolder_id > config_file.json
   ```

3. Open the configuration file in an editor, make your changes, and save the file.

   Note: When **aswatchfolderadmin** returns the JSON configuration, it obfuscates the password for the host with asterisks (******). If you do not want to update the password, leave it obfuscated (as asterisks) in the new file and the old password is used. To update the password, enter the new string. If no password is specified, then the password value is empty and transfers cannot be authenticated.

4. Update the Watch Folder configuration by sending the updated configuration file:

   ```
   # curl -kv --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X PUT
   -d @/path_to_json https://host:node_api_port/v3/watchfolders/watchfolder_id
   ```

Note: The header "X-aspera-WF-version:2017_10_23" is required when submitting POST, PUT, and GET requests to /v3/watchfolders on servers that are version 3.8.0 or newer. This enables Watch Folders to parse the JSON "source" and "target" objects in the format that was introduced in version 3.8.0.

For example:

```
# curl -kv --user watchfolder_admin:XF324cd28 -H "X-aspera-WF-version:2017_10_23" -X PUT -d
@/tmp/wf_config_update.json https://198.51.100.22:9092/v3/watchfolders/b394d0ee-1cda-4f0d-
b785-efdc6496c585
```

If the update is successful, then the following is returned:

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
```

Moving a Watch Folder from one user or daemon to another

To move a Watch Folder configuration, you must first retrieve the Watch Folder's configuration, make the desired changes, and then create a new Watch Folder with the modified configuration file. Follow the steps provided previously to query and save a configuration for the Watch Folder.

Open the configuration file in an editor and make the following changes:

1. Remove the "id" field.
2. Remove the "version" field.
3. Re-enter the password in the "pass" field.
4. Set proper watchfolderd IDs in the ("wfd_id") fields

Save the configuration file and then run the following command, specifying the modified configuration file as the JSON file:

```
# curl -k --user node_api_user:node_api_password -H "X-aspera-WF-version:2017_10_23" -X POST -d
@path/to/json_file https://host:node_api_port/v3/watchfolders
```

For example, to change the user to admin2, run the following:

```
# curl -k --user admin2:XF324cd28 -H "X-aspera-WF-version:2017_10_23" -X POST -d @~/
watchfolder_conf.json https://198.51.100.22:9092/v3/watchfolders
{
"id": "b394d0ee-1cda-4f0d-b785-efdc6496c585"
}
```

To verify that the configuration was updated, retrieve the configuration file again and look for your changes.

Deleting a Watch Folder

To remove a Watch Folder, run the following command:

```
# curl -sk --user node_api_user:node_api_password -X DELETE https://host:node_api_port/v3/
watchfolders/watchfolder_id
```

For example:

```
# curl -k --user watchfolder_admin:XF324cd28 -X DELETE https://198.51.100.22:9092/v3/
watchfolders/b394d0ee-1cda-4f0d-b785-efdc6496c585
```

To verify that the Watch Folder was removed, retrieve the list of Watch Folders with the command as shown previously. If the Watch Folder ID is no longer listed, the Watch Folder was successfully deleted.

Configuring Custom Watch Folder Permissions Policies

By default, users are not allowed to perform any Watch Folders-related actions, unless they are configured with admin ACLs. If you do not want every user to have admin permissions, configure users with customized permissions policies, including whether they are allowed or denied permission to create Watch Folders, create Watch and Watch Folder services, and edit policies. The policy is a JSON object that is assigned to specific users. Users can be assigned to multiple policies to incrementally allow or deny permissions.

Create a Permission Policy

Run the following command:

```
# curl -k --user node_api_user:node_api_password -X POST -d @path/to/json_file https://
localhost:9092/access_control/policies
```

Where the JSON file contains the permissions policy, as described in the next section. The Node API user must have permission to create policies to run this command.

Policy Syntax

A permissions policy is a JSON object with the following syntax:

```
{
   "id": "policy_name",
   "statements": [
      {
         "effect": "effect_value",
         "actions": [
            "permission_1",
            "permission_2",
            ...
            "permission_n"
         ],
         "resources": [
            "resource_id"
         ]
      }
   ]
}
```

The placeholders take the following values:

- *policy_name*: A descriptive name for the policy, such as "only-wfd-aspera". If no value is specified, a UUID is generated and returned in the output when the policy is created.
- *effect_value*: Set to `ALLOW` or `DENY`.
- *permission*: An action that the user is allowed or denied, depending on *effect_value*. Values can use * to match any sequence of characters. For example, to allow all Watch Folder-related actions, enter `"WF_*"`. See the following section for a complete list of permissions.
- *resource_id*: For Watch Folder-related permissions, specify the resources to which the actions apply by their Aspera Resource Name (ARN), using the following general syntax:

  ```
  arn:service:resource_type:resource
  ```

  Where *service* identifies the product (`watchfolder` or `watch`), *resource_type* is the type of resource (`wfd` for a Watch Folder daemon , `wf` for a Watch Folder), and *resource* is the resource ID, or a series of IDs to specify the daemon and Watch Folder ID of a specific Watch Folder. See the following section for examples.

Actions

The following actions are permissions to create, delete, and view policies, and assign users to policies. These actions do not require that you specify a value for `"resources"`. To allow all permissions, use `"PERM_*"`.

```
PERM_CREATE_POLICY
PERM_DELETE_POLICY
PERM_LIST_POLICIES
PERM_ATTACH_USER_POLICY
PERM_DETACH_USER_POLICY
PERM_LIST_USER_POLICIES
```

The following actions create, delete, and view Watch and Watch Folder services. These actions do not require that you specify a value for "resources". Users without these permissions must create Watch Folders that use existing Watch and Watch Folder services.

```
PERM_LIST_RESOURCES
PERM_CREATE_RESOURCE
PERM_DELETE_RESOURCE
```

The following actions create and delete Watch Folders. These actions require that you specify the wfd resource, as arn:watchfolder:wfd:*daemon*. To allow actions on Watch Folders as any daemon, use arn:watchfolder:wfd:*.

```
WF_CREATE_WATCHFOLDER
WF_DELETE_WATCHFOLDER
```

Note: Node API users must have PERM_LIST_RESOURCES allowed in order to allow WF_CREATE_WATCHFOLDER or WF_DELETE_WATCHFOLDER.

The following actions retrieve Watch Folder configuration and state, update the Watch Folder, and retry a Watch Folder drop. These actions require that you specify the wf resource, as arn:watchfolder:wf:*daemon*:*watchfolder_id*. To allow actions on any Watch Folders run by any daemon, use arn:watchfolder:wf:*:*.

```
WF_GET_WATCHFOLDER
WF_GET_WATCHFOLDER_STATE
WF_UPDATE_WATCHFOLDER
WF_RETRY_DROP
```

To allow all Watch Folder actions on all Watch Folders, enter "WF_*" as the action and "arn:watchfolder:wfd:*" as the resource.

Sample Policies

Allow the user to view policies and user permissions:

```
{
   "id": "read-permissions",
   "statements": [
      {
         "effect": "ALLOW",
         "actions": [
            "PERM_LIST_*"
         ],
         "resources": []
      }
   ]
}
```

Allow the user to do all Watch Folders actions:

```
{
  "id": "all-watch-folders",
  "statements": [
    {
      "effect": "ALLOW",
      "actions": [
        "WF_*",
        "PERM_LIST_RESOURCES"
      ],
      "resources": [
```

```
        "arn:watchfolder:wfd:*"
      ]
    }
  ]
}
```

Assigning Node API Users to Policies

Assign a user to one or more policies by running the following command:

```
# curl -k --user node_api_user:node_api_password -X PUT -d {"policies":["policy_id1",
"policy_id2"]} https://localhost:9092/access_control/users/username/policies
```

You can also assign a policy to multiple users at once:

```
# curl -k --user node_api_user:node_api_password -X PUT" -d {"users":["user1", "user2"]}
https://localhost:9092/access_control/policies/policy_id/users
```

To retrieve the IDs of available permissions policies, run the following command:

```
# curl -k --user node_api_user:node_api_password -X GET https://localhost:9092/access_control/
policies
```

To view the permissions policies that are assigned to a user, run the following command:

```
# curl -k --user node_api_user:node_api_password -X GET https://localhost:9092/access_control/
users/username/policies
```

To view the users that are assigned to a permissions policy, run the following command:

```
# curl -k --user node_api_user:node_api_password -X GET https://localhost:9092/access_control/
policies/policy_id/users
```

Editing Policies

To edit a policy, create a JSON configuration file as if you were creating a new policy, but do not include the "id". Run the following command to update the policy:

```
# curl -k --user node_api_user:node_api_password -X PUT -d @path/to/json_file https://
localhost:9092/access_control/policies/policy_id
```

To retrieve the configuration of an existing policy, run the following command:

```
# curl -k --user node_api_user:node_api_password -X GET https://localhost:9092/access_control/
policies/policy_id
```

Note: The policy name (`"id"`) cannot be edited. To change the name, create a new policy.

Updating the Docroot or Restriction of a Running Watch Folder Service

If **aswatchfolderadmin** returns the error code `err=28672` when you try to create a Watch Folder, confirm that the user's docroot or restriction allows access to the source directory specified in the JSON configuration file. You might have specified a destination that is not permitted by the docroot or restriction of the user running `asperawatchfolderd`, or you may have no docroot configured at all.

About this task

These instructions describe how to retrieve the docroot or restriction configuration for the user and update the docroot or restriction, if necessary. The configuration change automatically triggers asperawatchd that is associated with the user to restart.

Procedure

1. Run the following command to retrieve the docroot or restriction setting for the user:

```
# /opt/aspera/bin/asuserdata -u username | grep "absolute"
```

```
# /opt/aspera/bin/asuserdata -u username | grep "restriction"
```

- If no docroot is configured for the user, no output is returned. Proceed to the next step to set a docroot or restriction.
- If a docroot is configured, the command returns output similar to the following:

```
canonical_absolute: "/"
absolute: "/"
```

- If a restriction is configured, the command returns output similar to the following:

```
file_restriction: "file:////*"
```

If the user's docroot or restriction does not permit access to the source folder, proceed to the next step to update the docroot.

2. Configure a docroot or file restriction for the user.

   Docroots and path restrictions limit the area of a file system or object storage to which the user has access. Users can create Watch Folders and Watch services on files or objects only within their docroot or restriction.

   Note: Users can have a docroot or restriction, but not both or Watch Folder creation fails.

   To set up a docroot from the command line, run the following command:

```
# asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
```

   Restrictions must be set from the command line:

```
# asconfigurator -x "set_user_data;user_name,username;file_restriction,|path"
```

   The restriction path format depends on the type of storage. In the following examples, the restriction allows access to the entire storage; specify a bucket or path to limit access.

Storage Type	Format Example
local storage	For Unix-like OS: • specific folder: `file:////folder/*` • drive root: `file:////*` For Windows OS: • specific folder: `file:///c%3A/folder/*` • drive root: `file:///c*`
Amazon S3 and IBM Cloud Object Storage - S3	`s3://*`
Azure	`azu://*`
Azure Files	`azure-files://*`
Azure Data Lake Storage	`adl://*`
Alibaba Cloud	`oss://*`
Google Cloud	`gs://*`
HDFS	`hdfs://*`

   With a docroot or restriction set up, the user is now an Aspera transfer user. Restart asperanoded to activate your change:

Run the following commands to restart asperanoded:

```
# systemctl restart asperanoded
```

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

```
# service asperanoded restart
```

Creating, Managing, and Configuring Services

The asperarund manages both asperawatchd and asperawatchfolderd. It stores both their configurations in its database, automatically starts the services when they are added, and restarts services if they fail. It also enables admins to start services under different users without switching between accounts, and to apply logging and database configurations to all services.

Similar to other Aspera services, asperarund starts automatically upon installation and runs as a system daemon (asperarund).

Starting asperarund

If asperarund is not running, then you cannot create Watch Folders or start a watch. The service is started automatically during installation, but you might have to start it if it was disabled or stopped.

Configuring Services

Configuration settings for asperarund, asperawatchd, and asperawatchfolderd are located in the <server> section of aspera.conf.

To view current service settings, run the following command and look for settings that start with rund, watch, watchd, and watchfolderd:

```
# /opt/aspera/bin/asuserdata -a
```

For more information on configuring , see:

“Watch Service Configuration” on page 235
“Watch Folder Service Configuration” on page 195

Configuring asperarund

Logging and the Redis database used by asperarund is configured in aspera.conf:

```
<server>
    ...
    <rund>
        <log_level>log</log_level>
        <log_directory>AS_NULL</log_directory>
        <db_spec>redis:127.0.0.1:31415</db_spec>
    </rund>
    <watch>
        ...
    </watch>
</server>
```

Run the corresponding **asconfigurator** command to edit a setting:

```
# asconfigurator -x "set_server_data;rund_log_level,log_level"
# asconfigurator -x "set_server_data;rund_log_dir,path"
# asconfigurator -x "set_server_data;rund_db_spec,db_spec"
```

Setting	Description	Default
log_level	The level of detail for asperarund logging. Valid values are log, dbg1, and dbg2.	log

Setting	Description	Default
`log_directory`	Log to the specified directory.	The Aspera logging file (“Log Files” on page 356).
db_spec	Use the specified Redis database, which is defined with the syntax `redis:`*`ip_address`*`:`*`port`*.	`redis:127.0.0.1:31415` (the localhost on port 31415).

Creating asperawatchd and asperawatchfolderd

Both asperawatchd and asperawatchfolderd run under system users. These users must have a docroot configured for them in `aspera.conf` and have write permissions to the default log directory if no custom log directory is configured in `aspera.conf`. Aspera recommends running asperawatchd under root, and selecting a user to run asperawatchfolderd as described in “Choosing User Accounts to Run Watch Folder Services” on page 184. For more information, see “Starting Aspera Watch Services and Creating Watches” on page 233 and “Creating a Push Watch Folder with aswatchfolderadmin” on page 187.

To start asperawatchd and asperawatchfolderd, run the corresponding command:

```
# /opt/aspera/sbin/asperawatchd --user username
# /opt/aspera/sbin/asperawatchfolderd --user username
```

A Watch service must be running under a user before a Watch Folders service can be created for that user.

Managing Services

Use the **asrun** command line utility to view, enable, disable, or delete services.

The general syntax of **asrun** commands is:

```
# /opt/aspera/bin/asrun send [options]
```

Run `asrun send -h` to output a complete list of options.

View a list of running services

```
# /opt/aspera/bin/asrun send -l
```

The output is similar to the following:

```
[asrun send] code=0
{
    "services": [
         {
              "id":"52ca847a-6981-47e1-9f9b-b661cf298af1",
              "configuration": {
                    "enabled":true,
                    "run_as": {
                         "pass": "*****",
                         "user":"root"
                     },
                     "type":"WATCHD"
               },
               "state":"RUNNING",
               "state_changed_at":"2016-10-20T19:14:34Z"
          },
         {
               "id":"d109d1bd-7db7-409f-bb16-ca6ff9abb5f4",
               "configuration": {
                     "enabled":true,
                     "run_as":{
                          "pass": "*****",
                          "user":"root"
                     },
                     "type":"WATCHFOLDERD"
               },
               "state":"RUNNING",
               "state_changed_at":"2016-10-20T00:11:19Z"
           }
```

```
    ]
}
```

The Watch Service configuration includes the string `"type":"WATCHD"` and, before this entry in the output, a value for `"id"`. The Watch Folder service includes the string: `"type":"WATCHFOLDERD"`.

Disable a Service

Disabling a service stops the service but saves its configuration in the database. Disabled services can be restarted (enabled).

For example, to disable the `asperawatchfolderd` service with `"id":"d109d1bd-7db7-409f-bb16-ca6ff9abb5f4"`:

```
# /opt/aspera/bin/asrun send --disable="d109d1bd-7db7-409f-bb16-ca6ff9abb5f4"
[asrun send] code=0
null
```

Enable a Service

Enabling a stopped service starts the service. This command can be used to restart a service that stops due to an error, without changing the configuration to trigger a reload of the configuration.

For example, to enable the `asperawatchfolderd` service with `"id":"d109d1bd-7db7-409f-bb16-ca6ff9abb5f4"`:

```
# /opt/aspera/bin/asrun send --enable="d109d1bd-7db7-409f-bb16-ca6ff9abb5f4"
[asrun send] code=0
null
```

Delete a Service

Stop a service and remove its configuration from the database. A deleted service cannot be re-enabled.

Note: When deleting the `asperawatchfolderd` service, all existing Watch Folders started with that service are also deleted.

For example, to delete the `asperawatchfolderd` service with `"id":"d109d1bd-7db7-409f-bb16-ca6ff9abb5f4"`:

```
# /opt/aspera/bin/asrun send --delete="d109d1bd-7db7-409f-bb16-ca6ff9abb5f4"
[asrun send] code=0
null
```

The Aspera Watch Service

Automatically detect file system changes with the Aspera Watch Service.

Starting Aspera Watch Services and Creating Watches

The Aspera Watch Service (**asperawatchd**) is a file system change detection and snapshot service that is optimized for speed, scale, and distributed sources. On file systems that have file system notifications, changes in source file systems (new files and directories, deleted items, and renames) are detected immediately, eliminating the need to scan the file system. On file systems without file notifications, such as object storage, Solaris, and AIX file system scans are automatically triggered.

About this task

The Aspera Watch Service can be used on any local or shared (CIFS, NFS) host. However, when watching mounted shared storage and the change originates from a remote server, the Watch Service does not receive file notifications. In such cases, set <scan_period> in `aspera.conf` to frequent scans, such as 1 minute. See the following steps for instructions.

When used in conjunction with **ascp** commands, the Aspera Watch Service enables fast detection and transfer of new and deleted items. For more information on using watches with **ascp**, see “Transferring and Deleting Files with the Aspera Watch Service” on page 238.

To start the Aspera Watch Service and subscribe to (create) a watch:

Procedure

1. Configure a docroot or file restriction for the user.

   Docroots and path restrictions limit the area of a file system or object storage to which the user has access. Users can create Watch Folders and Watch services on files or objects only within their docroot or restriction.

   Note: Users can have a docroot or restriction, but not both or Watch Folder creation fails.

   To set up a docroot from the command line, run the following command:

   ```
   # asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
   ```

   Restrictions must be set from the command line:

   ```
   # asconfigurator -x "set_user_data;user_name,username;file_restriction,|path"
   ```

   The restriction path format depends on the type of storage. In the following examples, the restriction allows access to the entire storage; specify a bucket or path to limit access.

Storage Type	Format Example
local storage	For Unix-like OS: • specific folder: `file:////folder/*` • drive root: `file:////*` For Windows OS: • specific folder: `file:///c%3A/folder/*` • drive root: `file:///c*`
Amazon S3 and IBM Cloud Object Storage - S3	`s3://*`
Azure	`azu://*`
Azure Files	`azure-files://*`
Azure Data Lake Storage	`adl://*`
Alibaba Cloud	`oss://*`
Google Cloud	`gs://*`
HDFS	`hdfs://*`

   With a docroot or restriction set up, the user is now an Aspera transfer user. Restart asperanoded to activate your change:

   Run the following commands to restart `asperanoded`:

   ```
   # systemctl restart asperanoded
   ```

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

   ```
   # service asperanoded restart
   ```

2. Ensure the user has permissions to write to the default log directory if no directory is specified.

   For more information about configuring log directories, see“Watch Service Configuration” on page 235.

3. Configure Watch Service settings.

Though the default values are already optimized for most users, you can also configure the snapshot database, snapshot frequency, and logging. For instructions, see “Watch Service Configuration” on page 235.

4. Start a Watch Service under the user.

   The following command adds the Watch Service run under the user to the Aspera Run Service database:

   ```
   # /opt/aspera/sbin/asperawatchd --user username [options]
   ```

5. Verify that the Watch Service daemon is running under the user.

   Use the **aswatchadmin** utility to retrieve a list of running daemons. Daemons are named for the user who runs the service. For example, if you started a Watch Service under root, you should see the root daemon listed when you run the following command:

   ```
   # /opt/aspera/bin/aswatchadmin query-daemons
   [aswatchadmin query-daemons] Found a single daemon:
       root
   ```

6. Create a watch.

   A watch is a path that is watched by the Aspera Watch Service. To create a watch, users subscribe to a Watch Service and specify the path to watch. run the following command, where *daemon* is the username used to start the `asperawatchd` service and *filepath* is the directory to watch:

   ```
   # /opt/aspera/bin/aswatchadmin subscribe daemon filepath
   ```

   When you create a new subscription, you can also set watch-specific logging, database, scan period, and expiration period, and override `aspera.conf` settings.

   Note: The default scan period is 30 minutes. If you are watching a file system that does not support file system notifications (such as object storage, mounted storage (NFS), Solaris, and AIX), Aspera recommends setting a more frequent scan to detect file system changes quicker.

   For more information on using these options, see “Managing Watch Subscriptions” on page 237 or run:

   ```
   # /opt/aspera/bin/aswatchadmin subscribe -h
   ```

   Note: The default expiration for watches is 24 hours. If a watch subscription expires before the user resubscribes to it, a new subscription must be created.

Watch Service Configuration

The Aspera Watch Service configuration in the `<server>` section of `aspera.conf` includes the snapshot database, snapshot frequency, and logging:

```
<server>
    <rund>...</rund>
    <watch>
        <log_level>log</log_level>
        <log_directory>AS_NULL</log_directory>
        <db_spec>redis:host:31415:domain</db_spec>
        <watchd>
            <max_directories>1000000</max_directories>
            <max_snapshots>10000</max_snapshots>
            <snapshot_min_interval>3s</snapshot_min_interval>
            <snapshot_min_changes>100</snapshot_min_changes>
            <scan_threads>16</scan_threads>
        </watchd>
        <watchfolderd>...</watchfolderd>
    </watch>
</server>
```

To view current settings without opening `aspera.conf`, run the following command and look for settings that start with `watch` and `watchd`:

```
# /opt/aspera/bin/asuserdata -a
```

Note: Logging and database settings apply to both the Watch Service and Watch Folders services.

Configuring Watch Service Settings

Configure the Watch Service by using **asconfigurator** commands with this general syntax:

```
# /opt/aspera/bin/asconfigurator -x "set_server_data;option,value"
```

Options and values are described in the following table.

Configuration Options and Values

asconfigurator option **aspera.conf setting**	**Description**	**Default**
watch_log_dir <log_dir>	Log to the specified directory. This setting applies to both the Watch Service and Watch Folders services.	The Aspera logging file (“Log Files” on page 356).
watch_log_level <log_level>	The level of detail for Aspera Watch Service logging. This setting applies to both the Watch Service and Watch Folders services. Valid values are `log`, dbg1, and dbg2.	`log`
watch_db_spec <db_spec>	Use the specified Redis database, which is defined with the syntax `redis:`*`ip_address`*`:`*`port`*`[:`*`domain`*`]`. This setting applies to both the Watch Service and Watch Folders services.	`redis:127.0.0.1:31415`
watchd_max_directories <max_directories>	The maximum number of directories that can be watched (combined across all watches). This setting is used only on Linux machines to overwrite the system value `/proc/sys/fs/inotify/max_user_watches`. To overwrite the system value with the `aspera.conf` value, run the setup procedure in the admin tool: `# aswatchadmin setup`	1000000
watchd_max_snapshots <max_snapshots>	The number of snapshots that are stored in the database before the oldest are overwritten.	10000
watchd_snapshot_min_interval <snapshot_min_interval>	The maximum amount of time between snapshots. If this period passes without the minimum number of changes to trigger a snapshot, a new snapshot is taken.	3s
watchd_snapshot_min_changes <snapshot_min_changes>	The minimum number of changes that trigger a snapshot. If this number is reached before the snapshot minimum interval passes, a new snapshot is taken.	100
watchd_scan_threads <scan_threads>	The number of threads to use to scan the watched folder. More threads increase the speed of the scan, particularly for folders with large numbers of files, but require more of your computer's resources.	16

Setting Custom Watch Scan Periods

When a new subscription to a watch is created, it uses the default scan period of 30 minutes unless otherwise specified. You can also modify the scan period of an existing subscription.

Set the Default Scan Period When Upgrading from 3.7.4 or earlier to 3.8.0 or later

To update the default scan period that is applied during the migration, run the following command:

```
# asconfigurator -x "set_server_data;watchd_scan_period,value"
```

Modify the Scan Period of an Existing Subscription

In the subscription model, you cannot update the scan period of an existing subscription. Instead, create a new subscription and let the old one expire. To retrieve the configuration of the existing subscription, run the following command:

```
# /opt/aspera/bin/aswatchadmin query-subscription daemon
```

To create a new subscription and set the scan period, run the following command:

```
# /opt/aspera/bin/aswatchadmin subscribe daemon path --scan-period=seconds
```

Managing Watch Subscriptions

The Aspera Watch Service can watch the entire area of the file system to which the user has access. Individual watches are created by subscribing to the service and specifying a portion of the file system to watch. Each subscription can specify a scan period, database, and subscription expiration. When subscriptions overlap in the file system, the shortest scan period is used to scan the shared area.

Watch subscriptions are managed by using the **`aswatchadmin`** command line utility.

Create a new subscription to a watch

```
# /opt/aspera/bin/aswatchadmin subscribe daemon filepath [options]
```

Options include:

`--db-spec=`*`type`*`:`*`host`*`:`*`port`*
: Use the specified Redis database, which is defined with the syntax `redis:`*`ip_address`*`:`*`port`*`[:`*`domain`*`]`.

`-L, --logdir=`*`log_dir`*
: Specify the location for watch logging, particularly if the user does not have access to the default log location (`/var/log/aspera.log`) or the location specified in `aspera.conf`.

`--scan-period=`*`seconds`*
: The time between file system scans of the watches (from end of one to start of the next). These scans are independent of the snapshot minimum interval and snapshot minimum changes to ensure that changes are identified. To never scan (asperawatchd relies entirely on file notifications), set to "infinite".

  On file systems without file notifications, such as object storage, mounted storage (NFS), Solaris, and AIX file system scans triggered by the scan period are used to detect file changes. In this case, set the scan period to frequently scan for changes. On operating systems that support file notifications (Linux, Windows, macOS), asperawatchd uses the file notifications as the primary means for detecting changes, and the scan period serves as a backup. In this case, the default value of 30 minutes is usually acceptable and no change is necessary. To never scan, and rely entirely on file notifications, set to `infinite`.

-x, --expire_in=*seconds*
How long the watch subscription lasts before being removed from the database, with a default of 86400 seconds (24 hours). Users can resubscribe to a watch before the expiration period. Once a watch subscription expires, a new subscription must be created.

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

List subscriptions for a specific daemon

```
# /opt/aspera/bin/aswatchadmin query-subscriptions daemon
```

The output includes the subscription IDs, which are used to unsubscribe and resubscribe to the specific watch.

Unsubscribe from a watch

```
# /opt/aspera/bin/aswatchadmin unsubscribe daemon subscription_id
```

Resubscribe to a watch

```
# /opt/aspera/bin/aswatchadmin resubscribe daemon subscription_id
```

Transferring and Deleting Files with the Aspera Watch Service

When used in conjunction with **ascp** commands, the Aspera Watch Service (**asperawatchd**) allows for fast detection and sending of new and deleted items. By comparing snapshots of the file directory it is watching, **asperawatchd** generates file lists for **ascp** transfers.

Prerequisites

To generate snapshots and file lists, configure and start **asperawatchd**. For more information, see Configuring the Aspera Watch Service.

Creating a Subscription, Snapshots, and Snapshot Differential

Procedure

1. Create a subscription and decide how to manage its expiration.

   ```
   # /opt/aspera/bin/aswatchadmin subscribe daemon filepath [options]
   ```

   By default, subscriptions expire in 24 hours. If your snapshot comparisons will be spaced more than 24 hours apart, either set the expiration time to a duration longer than the time between snapshots (add `--expire_in=`*seconds* to the command) or send a resubscribe command periodically to maintain the subscription.

   For more information on creating subscriptions and resubscribing to them, see “Managing Watch Subscriptions” on page 237.

   In the following example, user **aspera** subscribes to **/projectA/source** and the subscription expires in 48 hours:

   ```
   # /opt/aspera/bin/aswatchadmin subscribe aspera /projectA/source --expire_in=172800
   [aswatchadmin subscribe] Successfully created subscription {"identifier":"bec581b3-3c34-47d7-
   a719-93f26f8272d1","path":"file:////projectA/source","scan_period":
   {"sec":9223372036854775807,"usec":999999},"expiration":"2018-03-15T07:39:21Z"}
   ```

   Record the subscription ID (the value of "identifier" in the output) for use in creating the snapshot. You can also retrieve the subscription ID later.
2. Create a snapshot.

   ```
   # /opt/aspera/bin/aswatchadmin create-snapshot daemon subscription_id
   ```

If you do not have the subscription ID, run the following command:

```
# /opt/aspera/bin/aswatchadmin query-subscriptions daemon
```

In the following example, user **aspera** creates a snapshot of the directory that is watched by subscription **bec581b3-3c34-47d7-a719-93f26f8272d1**:

```
# /opt/aspera/bin/aswatchadmin create-snapshot aspera bec581b3-3c34-47d7-a719-93f26f8272d1
[aswatchadmin create-snapshot] Successfully created snapshot 1.
```

3. After the desired interval, create another snapshot to compare with the previous snapshot.

   The snapshot ID is automatically incremented with each `create-snapshot` command. For example, running the same command as the previous step outputs a new snapshot:

   ```
   # /opt/aspera/bin/aswatchadmin create-snapshot aspera bec581b3-3c34-47d7-a719-93f26f8272d1
   [aswatchadmin create-snapshot] Successfully created snapshot 2.
   ```

4. Generate the snapshot differential between the most recent snapshot and the snapshot before it.

   To create a snapshot differential that outputs a list that can be used by **ascp**, run the following command:

   ```
   # /opt/aspera/bin/aswatchadmin snapshot-differential daemon subscription_id snapshot_id --
   format=PATH
   ```

   Where the snapshot ID is the latest snapshot. For example:

   ```
   # /opt/aspera/bin/aswatchadmin snapshot-differential aspera bec581b3-3c34-47d7-
   a719-93f26f8272d1 2
   /new_file.png
   /new_file.pdf
   ```

   Save the file list for use in the transfer session.

5. Send the new and modified files with **ascp** or **ascp4**.

   Use the `--source-prefix` option to append the watch directory path to the filepaths in the list:

   ```
   # ascp --file-list=filelist_pathname --source-prefix=prefix --mode=send --user=username --
   host=host target_directory
   ```

   For example:

   ```
   # ascp --file-list=/Users/aspera/filelist.txt --source-prefix=/projectA/source --mode=send --
   user=aspera --host=10.0.0.1 /projectA/destination
   new_file.png       100%   10MB    9.7Mb/s     00:07
   new_file.pdf       100%   100MB   9.7Mb/s     00:35
   Completed: 112640K bytes transferred in 42 seconds
    (268190 bits/sec), in 2 files.
   ```

Results

Removing Files from the Target Directory

The **asdelete** utility compares the source directory with the target directory and deletes extraneous files from the target directory. Run first with the `-d` option to do a dry run and view a list of files that would be deleted in an actual run. If the initiator of the **asdelete** command is a Windows OS, files that contain ASCII characters (such as <, |, ?, or ") are not deleted and an error is logged.

CAUTION: asdelete follows symbolic links, which can result in files being deleted that are not within the target directory.

```
# /opt/aspera/bin/asdelete --host host --auth-name username --auth-pass password /
source_directory /target_directory
```

For example:

```
# /opt/aspera/bin/asdelete --host 10.0.0.1 --auth-name aspera --auth-pass !XF345lui@0 /projectA/
source /projectA/destination
```

View the target directory to confirm deletion of the correct files.

Aspera Sync

A complete guide to IBM Aspera Sync.

Introduction

Learn about the key features and capabilities of IBM Aspera Sync.

Note: Aspera Sync capabilities are only provided with the IBM Aspera Enterprise and IBM Aspera Endpoint products. Aspera Sync is not provided with, or supported by, the HSTS or HSTE standalone products.

Overview

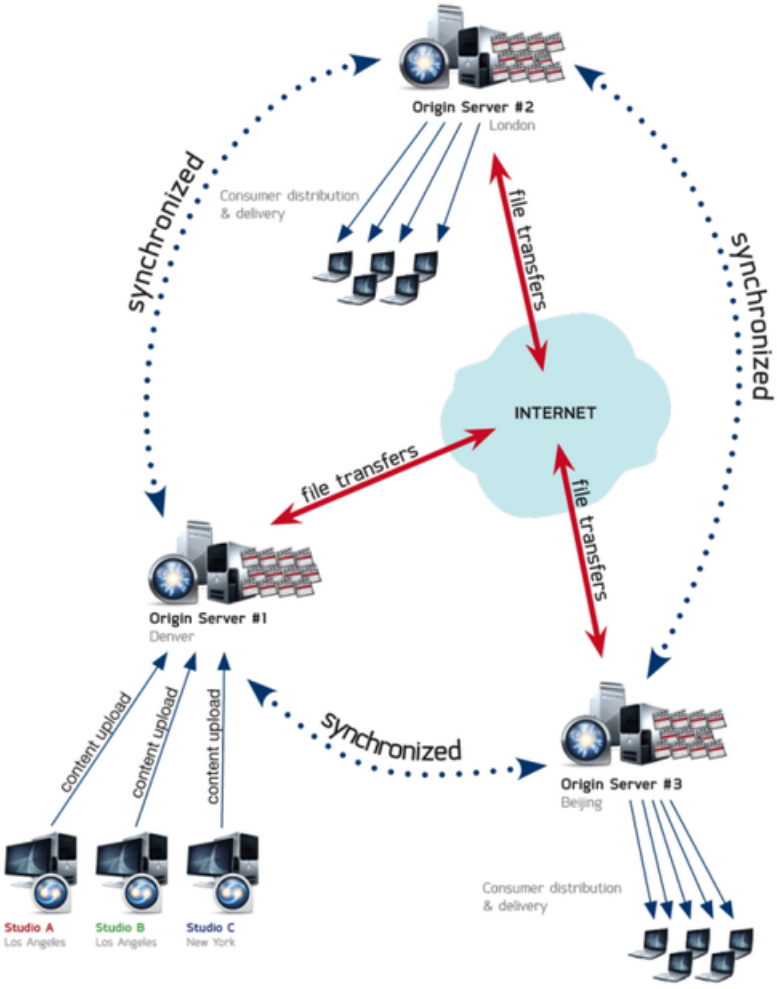
IBM Aspera Sync is a software application that provides high-speed, highly-scalable, multi-directional, file-based replication and synchronization. Syncis designed to fill the performance gap of uni-directional file synchronization tools like **rsync**, which are often slow for synchronizing large files and large sets of files over the WAN. Additionally, Sync surpasses the capability of uni-directional synchronization tools with full support for bi-directional synchronization.

Sync offers the following key capabilities:

- Utilizes high-speed Aspera FASP transport for moving data at maximum speed over the WAN, whereas traditional synchronization tools are built on TCP. Sync transfers new data between remote hosts at full bandwidth capacity, regardless of round-trip delay and packet loss, and does not degrade in performance for large file sizes.
- Compares against a local snapshot, thereby avoiding making a comparison against the remote file system over the WAN, which is used by most traditional tools and can be slow.
- Recognizes file system changes (such as moves and renames) on the source and propagates these changes to the destination. Traditional tools treat these operations as deletion of old data and then recreate or re-transfer the new data, which can lead to costly data copying over the WAN.
- Supports bi-directional and multi-directional synchronization topologies, where files are changing on multiple nodes. For a bi-directional synchronization, Syncruns with a bi-directional option. For a multi-directional synchronization, one session is run for each peer to remain sync. Any topology that has an acyclic graph topology between peers is supported.
- Uses file system notifications for change notification, when available.
- Monitors file contents and waits for files to be stable (no longer changing in md5sum) before transferring. The wait period is configurable and is designed to avoid transferring only partially complete files.

Sync is a command-line tool, **async**, that uses an SSH connection to establish connectivity with its remote peers and is spawned as an SSH subsystem binary on the remote system. The program can be run one time or periodically (through a cron tab scheduled job) on file systems that do not provide asynchronous change notification, or in a continuous mode on file systems that do support asynchronous change notification. Sync is designed to process files and transfer new data in a continuous pipeline for maximum speed, even when running in scan-only mode (when no file system change notification is available).

Sample Sync Deployment Diagram



Synchronization and Direction Modes

Sync offers two modes of operation: one-time ("on demand") synchronization and continuous synchronization, as well as three direction modes: uni-directional, bi-directional, and multi-directional.

One-time vs. Continuous Synchronization

One-time synchronization

In this mode, **async** performs synchronization of the endpoints, and exits. If available, **async** uses an existing snapshot to determine changes, unless specifically instructed to drop the snapshot and scan the file system again (see the `-x` option in “async Command Reference” on page 255).

This mode should be used for one_time operations, or for periodic, scheduled synchronizations where file systems do not support event-based change notification. For the latter, **async** can be scheduled as a cron job to run periodically.

One-time synchronization is supported between all operating systems.

Continuous synchronization

In this mode, Sync synchronizes the endpoints and continues running. As file system updates occur (for example, files or directories are added, deleted or modified), Sync detects these changes and synchronizes with the peer endpoint.

Continuous mode is supported only when the file source is Windows, Linux, or macOS. See the following table for the operating system requirements for the Sync server and client for the different Syncdirections.

Continuous Sync Direction	Supported Sync Client OS	Supported Sync Server OS
PUSH	Linux, Windows, macOS	All
PULL	All	Linux, Windows, macOS
BIDI	Linux, Windows, macOS	Linux, Windows, macOS

Sync Direction Modes

Uni-directional

Similar to **rsync**, the uni-directional mode supports replication of files and directories, and any updates to these (including deletions, renames, moves, and copies) from a source to a target. The direction of replication can be specified as a "push" or "pull" operation, relative to the initiating host. Once a snapshot is taken after the first replication, all file system updates are recognized against this snapshot, and no comparison of source to target over the WAN is performed (as in **rsync**). Sync supports most of the same uni-directional synchronization options as **rsync**, such as include/exclude filters, overwrite only if newer, symbolic link handling, and preservation of file system ownership and timestamps.

Bi-directional

Bi-directional mode supports the replication of all file and directory updates between the peers. For any case in which the most recent version of an update cannot be reliably determined, or when a file changes on both endpoints concurrently, Syncflags the update as a conflict and leaves the peer file systems in their present state (and in conflict). Files in conflict can be reviewed using the **asyncadmin** command-line tool (see “asyncadmin Command-Line Options” on page 284). In this version, it is up to the operator to resolve conflicts manually.

Multi-directional

Multi-directional synchronization requires one Sync session (one **async** process execution) for each remote peer. Any number of **async** processes can be run concurrently, and any number of peers can be synchronized concurrently; however, a downstream peer cannot be configured to synchronize "back" in a loop to an upstream peer.

Sync FAQ

Get answers about what Sync does and how it does it.

What does Sync actually do?

Sync synchronizes new and modified files and directories between remote endpoints. It moves, deletes, renames, and transfers new file contents as needed. For example:

- Moving a file out of the synchronized directory results in deletion at the remote peer.
- Moving a file into the synchronized directory results in a copy at the remote peer.
- Renaming a file in a previously synchronized directory renames the file at the remote peer; moving a file in a previously synchronized directory results in the same move operation at the peer.

How does Sync differ from rsync?

Sync is a high-speed replacement for **rsync** in uni-directional mode, and is designed to be a drop-in replacement with similar command-line options. Sync also supports bi-directional and multi-directional synchronization. The following key capabilities distinguish it from **rsync**:

- Uses Aspera's high-speed FASP transport technology, while rsync transfers over traditional TCP.
- Operates in push, pull and bi-directional modes.

- Circumvents the typically slower comparison of the local system to the remote system over the WAN, and instead, it efficiently compares the current file system state to a *snapshot* of the last sync.
- Detects and implements file or directory moves and renames to avoid unnecessary transfers over the network.
- Waits for the systems to become stable (that is, it detects whether files are still being modified) before performing synchronization.

For a comparison of **async** options versus **rsync** options, see rsync vs. async Uni-directional Example.

How is one-time mode different from continuous mode?

Sync offers two modes of operation: one-time ("on-demand") synchronization and continuous synchronization. When running in one-time mode, it synchronizes once and exits. In continuous mode, on the other hand, it offers constant synchronization between file systems.

Continuous mode can only be used where file system change notification (that is, *inotify*, which monitors file system events) is available on the systems that are running **async**. NFS-mounted file systems do not support *inotify* change notification for updates made by remote NFS clients, so in these scenarios, **async** should be run in one-time mode (which can be scheduled through cron). The Sync scan mode is designed for maximum speed and is fully pipelined with transfer, so as to allow for maximum performance even in one-time mode.

In what directions does Sync work?

Sync works in multiple directions: push, pull, and bi-directional.

- Sync supports pushing content from the local system to a remote system, and pulling content from a remote system to the local system.
- Bi-directional synchronization occurs between two endpoints, such that file system changes on either end (local or remote) are replicated on both sides.

How are conflicts handled in bi-directional mode?

A conflict situation can arise in bi-directional mode when a file or directory changes content, an entity is renamed before synchronization has completed, or the change occurs on both endpoints concurrently such that the "newer" version cannot be reliably determined. Sync reports such conflicts and does not modify either file system, leaving the file systems in conflict. For instructions on resolving conflicts, see “Resolving Bidirectional Sync File Conflicts” on page 288.

How much space is required for an Sync snapshot?

Snapshots require up to 1 GB of disk space for every 1 million files, and an additional 1 GB for cleanup purposes. For optimum performance, Aspera recommends that the file system have at least 2 GB free per 1 million files, and 3 GB free per 1 million files on Windows (due to the poor performance of Windows NTFS when more than half of the available disk space is occupied).

Sync Set Up

Sync is installed when you install your HSTS; your license must enable Sync. Before using Sync, prepare the file systems to synchronize and plan your replication strategy, as described in the following sections.

Configuring Sync Endpoints

Sync reads configuration settings from `aspera.conf`, which can be edited using **asconfigurator** commands or manually. The following sections provide instructions for setting Aspera-recommended security configuration, instructions for how to edit other configurations, a reference for many of the available configuration options, and a sample `aspera.conf`.

Aspera-Recommended Configuration

Aspera recommends setting the following configuration options for greatest security. Additional settings are described in the following table.

Note: To synchronize with AWS S3 storage, you must configure specific locations for the log and database directories. For more information, see “Synchronizing with AWS S3 Storage” on page 276.

1. Set the location for the Sync log for each transfer user.

   By default, Sync events are logged to the system log (see “Logging” on page 285). Aspera recommends setting the log to a directory within the transfer user's home folder.

   Log location, size, and log level can be configured for both **ascp** and **async** by setting default or user-specific configurations in `aspera.conf`. For instructions, see “Server Logging Configuration for Ascp and Ascp4” on page 107.

   To set a logging directory for **async** that is separate from **ascp**, you can set `async_log_dir`. For example:

   ```
   # /opt/aspera/bin/asconfigurator -x "set_user_data;user_name,username;async_log_dir,log_dir"
   ```

   Note: If `async_log_dir` is not set, then the logging configuration for **ascp** is applied. The client can override the server logging settings with the `-R` option.

2. Set the location for the Sync database for each transfer user.

   Sync uses a database to track file system changes between runs of the same session (see “The Sync Database” on page 249). The Aspera Sync database should not be located on CIFS, NFS, or other shared file systems mounted on Linux, unless you are synchronizing through IBM Aspera Proxy. If server data are stored on a mount, specify a local location for the Sync database. Aspera recommends setting the database location to a directory within the user's home folder by using the same approach as setting the local Aspera Sync log:

   ```
   # /opt/aspera/bin/asconfigurator -x "set_user_data;user_name,username;async_db_dir,db_dir"
   ```

   This setting overrides the remote database directory specified by the client with the `-B` option.

   Note: If the transfer user's docroot is a URL (such as `file:////*`), then `async_db_dir` must be set in `aspera.conf`. For an example, see “Synchronizing with AWS S3 Storage” on page 276.

3. If the Sync source files are on a NFS or CIFS mount, create a mount signature file.

   Sync can use a mount signature file to recognize that the source is on a mount. If you do not use the mount signature file and the NFS or CIFS mount is unreachable, Sync considers those files as deleted and delete them from the other endpoint.

   To create a mount signature file, create the file in the parent directory of the source directory on the mount. For example, if the Sync directory is `/mnt/sync/data`, create the mount signature file `/mnt/sync/mount_signature.txt` by running the following command:

   ```
   # echo mount >> /mnt/sync/mount_signature.txt
   ```

   When you run a Sync session, use `--local-mount-signature=mount_signature.txt` if the local source is on a mount and `--remote-mount-signature=mount_signature.txt` if the remote source is on a mount. For bidirectional Sync sessions between mounts, use both.

Configuring Other Settings

To configure Syncsettings in `aspera.conf` by using **`asconfigurator`** commands, use the following general syntax for setting default values (first line) or user-specific values (second line):

```
# /opt/aspera/bin/asconfigurator -x "set_node_data;option,value"
# /opt/aspera/bin/asconfigurator -x "set_user_data;user_name,username;option,value"
```

To manually edit `aspera.conf`, open it in a text editor with administrative privileges from the following location:

`/opt/aspera/etc/aspera.conf`

See an example `aspera.conf` following the settings reference table. For an example of the asperawatchd configuration, see "Watch Service Configuration" on page 235.

After manually editing `aspera.conf`, validate that its XML syntax is correct by running the following command:

```
# /opt/aspera/bin/asuserdata -v
```

This command does not check if the settings are valid.

Sync Configuration Options

asconfigurator option **aspera.conf setting**	**Description and Value Options**
async_connection_timeout <async_connection_timeout>	The number of seconds `async` waits for a connection to be established before it terminates. Value is a positive integer. (Default: 20) If synchronization fails and returns connection timeout errors, which could be due to issues such as under-resourced computers, slow storage, or network problems, set the value higher, from 120 (2 minutes) to even 600 (10 minutes).
async_db_dir <async_db_dir>	Specify an alternative location for the async server's snap database files. If unspecified, log files are saved in the default location or the location that is specified by the client with the **-B** option.
async_db_spec <async_db_spec>	Value has the syntax `sqlite:`*`lock_style`*`:`*`storage_style`*. (Default: undefined) *`lock_style`*: Specify how **async** interfaces with the operating system. Values depend on operating system. Unix-based systems have the following options: • empty or `unix`: The default method that is used by most applications. • `unix-flock`: For file systems that do not support POSIX locking style. • `unix-dotfile`: For file systems that do not support POSIX nor flock-locking styles. • `unix-none`: No database-locking mechanism is used. Allowing a single database to be accessed by multiple clients is not safe with this option. *`storage_style`*: Specify where Syncstores a local database that traces each directory and file. Three values can be used: • undefined or `disk`: The default option. Read and write the database to disk. This provides maximum reliability and no limitations on the number of files that can be synchronized. • `lms`: The database is loaded from disk into memory at startup, changes during the session are saved to memory, and the database is saved to disk on exit. This option increases speed but all

asconfigurator option **aspera.conf setting**	**Description and Value Options**
	changes are lost if **async** stops abruptly, and the number of synchronized files is limited by available memory. • `memory`: The database is stored completely in memory. This method provides maximum speed but is not reliable because the database is not backed up to disk.
async_enabled <async_enabled>	Enable (set to `true`, default) or disable (set to `false`) Sync. When set to `false`, the client `async` session fails with the error `"Operation 'sync' not enabled or not permitted by license"`.
async_log_dir <async_log_dir>	Specify an alternative location for the async server's log files. If unspecified, log files are saved in the default location or the location that is specified by the client with the **-R** option. For information on the default log file location, see "Logging" on page 285.
async_log_level <async_log_level>	Set the amount of detail in the async server activity log. Valid values are `log` (default), dbg1, or dbg2.
async_session_timeout <async_session_timeout>	The number of seconds `async` waits for a non-responsive session to resume before it terminates. Value is a positive integer. (Default: 20)
directory_create_mode <directory_create_mode>	Specify the directory creation mode (permissions). If specified, create directories with these permissions irrespective of `<directory_create_grant_mask>` and permissions of the directory on the source computer. This option is applied only when the server is a Unix-based receiver. Value is a positive integer (octal). (Default: undefined)
directory_create_grant_mask <directory_create_grant_mask>	Specify the mode for newly created directories if `directory_create_mode` is not specified. If specified, directory modes are set to their original modes plus the grant mask values. This option is applied only when the server is a Unix-based receiver and when `directory_create_mode` is not specified. Value is a positive integer (octal). (Default: 755)
preserve_acls preserve_xattrs <preserve_acls> <preserve_xattrs>	Specify if the ACL access data (`acls`) or extended attributes (`xattrs`) from Windows or macOS files are preserved. Three modes are supported. (Default: none) `native`: `acls` or `xattrs` are preserved by using the native capabilities of the file system. If the destination does not support `acls` or `xattrs`, **async** generates an error and exits. `metafile`: `acls` or `xattrs` are preserved in a separate file. The file is in the same location and has same name, but has the added extension `.aspera-meta`. The `.aspera-meta` files are platform-independent, and files can be reverted to native form if they are synchronized with a compatible system. none: No `acls` or `xattrs` data is preserved. This mode is supported on all file systems.

asconfigurator option **aspera.conf setting**	**Description and Value Options**
	ACL preservation is only meaningful if both hosts are in the same domain. If a SID (security ID) in a source file does not exist at a destination, the sync proceeds but no ACL data is saved and the log records that the ACL was not applied. The `aspera.conf` settings for `acls` or `xattrs` can be overwritten by using the `--preserve-acls` or `--preserve-xattrs` options, respectively, in a command-line **async** session.

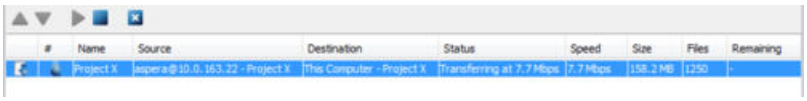
Example Sync Configuration in `aspera.conf`

```
<file_system>
   ...
   <directory_create_mode> </directory_create_mode>
   <directory_create_grant_mask>755</directory_create_grant_mask>
   <preserve_acls>none</preserve_acls>
   <preserve_xattrs>none</preserve_xattrs>
   ...
</file_system>
   ...
<default>
   ...
   <async_db_dir>  </async_db_dir>
   <async_db_spec>  </async_db_spec>
   <async_enabled>true</async_enabled>
   <async_connection_timeout>20</async_connection_timeout>
   <async_session_timeout>20</async_session_timeout>
   <async_log_dir>AS_NULL</async_log_dir>
   <async_log_level>log</async_log_level>
   ...
</default>
```

Viewing Sync Transfers in the Aspera GUI

The HSTS GUI shows **async**-initiated transfers if Sync is run on the machine (as client) by default, whereas server **async** transfers are not shown.

In the following example, the GUI shows transfers associated with a Syncjob in which the remote user, aspera, is pushing files to the server folder for Project X.



You can configure the server and client reporting to the Aspera GUI with the following options.

Server reporting:

Server reporting is disabled by default. To enable the server to report Aspera Sync-initiated transfers:

1. Run the following command on the server:

   ```
   # asconfigurator -x "set_node_data;async_activity_logging,true"
   ```

2. Restart asperanoded to activate your changes.

   Run the following commands to restart `asperanoded`:

   ```
   # systemctl restart asperanoded
   ```

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

   ```
   # service asperanoded restart
   ```

Client reporting:

Client reporting is enabled by default. To disable the client from reporting Aspera Sync-initiated transfers, run the following command on the client machine:

```
# asconfigurator -x "set_client_data;async_management_activity_logging,false"
```

You do not need to restart asperanoded for this change to take effect.

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: Ascp4 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
Client setting = copy (default for async)	Copy	Copy	Skip	Skip	Skip

	Server setting = create, follow (default)	Server setting = create	Server setting = follow	Server setting = follow_wide	Server setting = none
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"
```

For more information, see “aspera.conf - File System Configuration” on page 91.

Client Configuration

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

The Sync Database

Each **async** session creates a database (`snap.db`) that is stored on both the local (client) computer and the remote (server) computer. The database records the state of the file system at the end of the last **async** session, and the next time the session is run, the file system is compared to the database to identify changes.

Sync Database Location and Structure

Sync creates private directories (`.private-asp`) to store the database and in-progress transfers (a transfer cache for pending files).

The Sync database directory is stored on the local computer in the directory specified by the `-b` option in the command line, and on the remote computer in the directory set for `<async_db_dir>` in the server's `aspera.conf` (or set by the client with `-B` if no value is set on the server).

Note: The Sync database does not work on CIFS, NFS, or other mounted shared file systems; therefore, `-B` and `-b` must specify a directory on a file system physically local to the endpoint host.

Multiple **async** sessions can synchronize the same directory or specify the same database directory (`-b` or `-B`), so for each session `async` creates a subdirectory in `.private-asp` that is named with the session name specified by `-N`. To allow the session name to be used as a directory name, names can only use standard alphanumeric characters and "_" and "-" characters.

Each `async` session must have a unique name. If multiple sessions synchronize the same directory or specify the same database directory (`-b`/`-B`), then the session names *must* be unique. For example, you run an **async** session named **job1** that synchronizes the local directory `/data` and the remote directory `/data1`, and that stores the database in `/sync/db` on both endpoints. You cannot run another **async** session named **job1** that synchronizes `/data` with `/data2` and that stores the database in `/sync/db`; you must either run the session with a unique name or store the database in a different location.

Example 1: Bi-directional async

```
# -N ex1 -b /var/db -B /opt/aspera/var -d /data/users -r root@server:/storage/users -K bidi
```

The above command creates the following:

On the local computer (client):

- `/var/db/.private-asp/ex1/snap.db`
- `/data/users/.private-asp/ex1` (for transfer cache)

On the remote computer (server):

- `/opt/aspera/var/.private-asp/ex1/snap.db`
- `/storage/users/ex1` (for transfer cache)

Example 2: Uni-directional async

```
# -N ex2 -b /var/db -B /opt/aspera/var -d /data/users -r root@server:/storage/users -K push
```

The above command creates the following:

On the local computer (client):

`/var/db/.private-asp/ex2/snap.db`

On the remote computer (server):

`/opt/aspera/var/.private-asp/ex2/snap.db`

`/storage/users/ex2` (for transfer cache)

Changing Synchronization Direction Between Runs of the Same Session

Changing direction between runs of the same session is not supported. `async` fails with an error message and you must run it with `-x` (or `--reset`) or provide a new database directory.

Note: The `-x` or `--reset` options delete the existing database, and Sync must create a new one, which can take a long time if the file system contains many files and directories.

Starting a Sync Session When a Sync Database is Missing

If the database is missing or corrupted on either endpoint, repeating an `async` session fails with error messages similar to the following (in these examples, `/sync/peer` is the remote database directory and the session is named **push**):

```
Failed. Peer error: Local snapshot DB exists but remote snapshot DB /sync/peer/.private-asp/
push/snap.db does not exist
Failed. Peer error: file is encrypted or is not a database
Failed. Peer error: Corrupt database /sync/peer/.private-asp/push/snap.db
```

If this is the case, you can run `async` with `-x` or `--reset`. This option rebuilds the database, which can take some time for very large directories. A Sync session run with `--reset` has the following behavior:

1. If the private directory (`.private-asp`) is missing, Sync creates it.
2. If the database directory (`.private-asp/`*`session_name`*) is missing (and, therefore, the database file `snap.db` doesn't exist), Sync creates `snap.db` and its directory.
3. If the database directory does not contain the `snap.db` file, Synccreates it.

Deleting a Snapshot Database During Synchronization

Deleting either of the snapshot databases (client or server) that are in use by an active synchronization session results in undefined behavior. To recover, stop `async`, delete the database on the other side as well, and restart the session.

Running async

Sync uses the **async** command line tool to synchronize content from the source to the destination. **async** has many options for customizing the behavior of the synchronization, and this section describes how to compose an **async** session, the command line arguments, and examples for specific use cases.

Composing an Async Session

Sync has more than 80 options that can be used when composing an **async** session, but only a few are required, and Aspera recommends using several others.

About this task

For a complete list and descriptions of available options, see the "async Command Reference" on page 255. For configuration and option usage required to synchronize with AWS S3 storage, see "Synchronizing with AWS S3 Storage" on page 276.

Note: These instructions describe how to compose a bidirectional **async** session between a *Windows client* and a *Linux server*. They include the required and the recommended options in the correct order. You can use the short form or long form (POSIX) option tags. The complete set of commands using both tag formats are summarized at the end of the instructions.

Procedure

1. Confirm that both endpoints have Sync-enabled licenses and that the remote endpoint is running an Aspera transfer server application (HSTS or HSTE).

   Run **ascp -A** in the command line and look for `sync2` in the `Enabled settings` section.
2. Begin by invoking **async**.

   ```
   # async
   ```
3. Enter instance options.

   Instance options are used to configure the local (client) computer for the **async** session and are mostly optional. Aspera recommends that you include `-L` *`log_dir`* (or `--alt-logdir=`*`log_dir`*) to set client-side logging to a directory that you can access, because you might not have permission

to access the log in its default location (see “Logging” on page 285). The logging directory must not be in the directory that is being synchronized.

For example, if the Windows client's username is **morgana**, morgana can use `-L` to log to a directory in the home folder:

```
async -L "C:\Users\morgana\Aspera jobs\log"
```

In this example, the path must be in quotes because the path includes a folder name that contains a space. For more information on path formatting, see “async Command Reference” on page 255.

4. Name the session by using the `-N` option (or `--name=`*`pair`*).

   `-N` *`pair`* is required in **async** commands. The value for *pair* is a name that uniquely identifies the Aspera Sync session and is visible in IBM Aspera Console. `-N` *`pair`* must follow any instance options and must precede all session arguments. Names can only use standard alphanumeric characters, plus "_" and "-" characters.

   Note: If your remote host is an Aspera cluster, ensure that your session name is unique by naming the session with a descriptive string followed by the UUID of the local host, such as "cluster-sync-ba209999-0c6c-11d2-97cf-00c04f8eea45".

   For example, name the session **job1**:

   ```
   async -L "C:\Users\morgana\Aspera jobs\log" -N job1
   ```

   Once you name the session, you enter the session options. Session options define the transfer parameters including authentication, transfer rate and policy, database storage, and the folders to synchronize.

5. Provide authentication credentials.

   Sync supports three methods of authenticating to the server: SSH key, password, and basic token. Aspera recommends using SSH keys, unless your server requires a basic token.

   - **SSH key**: To use SSH key authentication, your SSH public key must be configured on the remote server. For instructions on creating keys and setting them up on the server, see the IBM Aspera High-Speed Transfer Server Admin Guide. Specify the path to your private key file by using the `-i` *`file`* (or `--private-key-path=`*`file`*) option.
   - **Password**: The password is the one associated with the Aspera transfer user account on the server. You can provide the password as an environment variable (ASPERA_SCP_PASS) or when prompted after starting the command.
   - **Basic token**: Basic tokens are used for synchronizing to Aspera products that require access key authentication, such as IBM Aspera on Cloud transfer service (AoCts) or IBM Aspera Transfer Cluster Manager (ATC Manager). For instructions on creating the basic token, see “Sync with Basic Token Authorization” on page 277. You can provide the token as an environment variable (ASPERA_SCP_TOKEN) or in the command line using the `-W` *`token_string`* (or `--token=`*`token_string`*) option.

   For example, use `-i` and specify the path to morgana's SSH private key in their home folder:

   ```
   async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa
   ```

   In this case, the path to the SSH key can use platform-agnostic path separators (/) and be entered without quotes around it because it does not have a space in it.

6. If the local data are stored on a mount or object storage, specify the locations for the local snapshot database.

   The snapshot database cannot be located on CIFS, NFS, or other shared file systems mounted on Linux. If the local files and directories specified in the previous step are on a mount, you must specify a local location using `-b` *`db_dir`* (or `--local-db-dir=`*`db_dir`*). The database must not be in the directory that is being synchronized.

For example, use `-b` to store the local snapshot database in morgana's "Aspera jobs" folder:

```
async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa -b
"C:\Users\morgana\Aspera jobs\db"
```

7. Set transfer parameters.

   The same transfer rate and transfer policy options that are used to control **ascp** transfers can be applied to **async** sessions. Aspera recommends setting a target rate that is based on your available bandwidth and system capabilities. Set the target (maximum) rate using `-l` *`rate`* (or `--target-rate=`*`rate`*).

   For example, use `-l` to set the target rate to 500 Mbps:

   ```
   async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa -b
   "C:\Users\morgana\Aspera jobs\db" -l 500m
   ```

8. Specify the local directory for synchronization.

   Enter the local directory using `-d` *`ldir`* (or `--local-dir=`*`ldir`*).

   For example, use `-d` to set the local directory to morgana's **data** folder:

   ```
   async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa -b
   "C:\Users\morgana\Aspera jobs\db" -l 500m -d c:/users/morgana/data
   ```

9. Specify the transfer username, remote host, and remote directory for synchronization.

   Unlike previous options for which one short option flag was equivalent to one long option flag, when specifying the username, remote host, and remote directory, the short flag option is the equivalent of one to three long option flags. For example, if the username is **morgana**, the remote host IP address is **10.0.0.1**, and the remote directory is **/data**, then the following options are equivalent to each other:

   ```
   -r morgana@10.0.0.1:/data
   --remote-dir=morgana@10.0.0.1:/data
   --user=morgana --remote-dir=10.0.0.1:/data
   --user=morgana --host=10.0.0.1 --remote-dir=/data
   ```

   If the name of your remote directory contains an "@", use the `--user` option so that the "@" is not treated specially in the argument for `--remote-dir`.

   For example, use `-r` to set the username, remote host, and remote directory:

   ```
   async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa -b
   "C:\Users\morgana\Aspera jobs\db" -l 500m -d c:/users/morgana/data -r morgana@10.0.0.1:/data
   ```

10. If a source directory is on an NFS or CIFS mount, require Syncto use the mount signature file.

    Warning: If you do not use the mount signature file and the NFS or CIFS mount is unreachable, Sync considers those files as deleted and deletes them from the other endpoint.

    If the local endpoint is on a NFS or CIFS mount and the Sync is push or bidirectional, use `--local-mount-signature`. If the remote endpoint is on a NFS or CIFS mount and the Sync is pull or bidirectional, use `--remote-mount-signature`.

11. Specify the locations for the remote Sync log and database.

    On the server, Sync logs to the default location (see “Logging” on page 285) if no location is specified for <async_log_dir> in the server's configuration file. Aspera recommends using `-R` (or `--remote-logdir`) to specify a logging location to which you have access. The location must be within your docroot on the server, unless you are synchronizing with AWS S3 object storage. `-R` is overridden by the server's configuration file. If you are restricted to aspshell on the server, you cannot use this option.

    Aspera also recommends using `-B` (or `--remote-db-dir`) to specify a location for the remote Sync database. As with the log file, the location must be within your docroot, it is overridden by

<async_db_dir> in the server's configuration file, and you cannot use this option if you are restricted to aspshell.

As on the local computer, the Sync log and database must not be in a directory that is being synchronized.

For example, to set the remote log and snapshot database files to morgana's home folder:

```
async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa -b
"C:\Users\morgana\Aspera jobs\db" -l 500m -d c:/users/morgana/data -r morgana@10.0.0.1:/
data -R /morgana/async/log -B /morgana/async/db
```

12. Specify the synchronization mode.

    Sync can be run in three modes:

    - `push`: The contents of *ldir* are synchronized to *rdir*, with the *ldir* content overwriting the *rdir* content, by default (unless the overwrite options are specified otherwise, such as to only overwrite if *rdir* is older, or never overwrite).
    - `pull`: The contents of *rdir* are synchronized to *ldir*, with the *rdir* content overwriting the *ldir* content, by default.
    - `bidi` (bi-directional): The contents of *ldir* and *rdir* are synchronized, with newer versions of files and directories overwriting older versions in either *ldir* or *rdir*, by default.

    To synchronize the remote folder with the local folder use `-K push` (or `--direction=push`).

    For example, use `-K bidi` to do a bidirectional sync:

    ```
    async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa -b
    "C:\Users\morgana\Aspera jobs\db" -l 500m -d c:/users/morgana/data -r morgana@10.0.0.1:/
    data -R /morgana/async/log -B /morgana/async/db -K bidi
    ```

13. Preserve file attributes.

    When a file or directory is transferred between computers, the file is written to the destination as the transfer user and the file modification time (and creation time on Windows) are reset. Most users prefer to preserve timestamps from the source to the destination by using the `-t` option.

    For example, use `-t` timestamps:

    ```
    async -L "C:\Users\morgana\Aspera jobs\log" -N job1 -i c:/users/morgana/.ssh/id_rsa -b
    "C:\Users\morgana\Aspera jobs\db" -l 500m -d c:/users/morgana/data -r morgana@10.0.0.1:/
    data -R /morgana/async/log -B /morgana/async/db -K bidi -t
    ```

    Note: When synchronizing between Unix-like operating systems, you can also preserve the user IDs (uid) and group IDs (gid) from the source to the destination by using the options `-u -j` (equivalent to `--preserve-uid --preserve-gid`).

    Extended file attributes and ACLs can also be preserved; see the "async Command Reference" on page 255. When using `--dedup`, file metadata preservation is supported for `copy`.

Results

Summary

The instructions created the following Syncsession, shown using short option flags and POSIX (long) flags. Each option is shown on a separate line for clarity, but should be entered in the command line as a single line.

Warning: This example does not include the option to make Sync check for a mount signature file. If a source is on a NFS or CIFS mount, include `--local-mount-signature` and `--remote-mount-signature` to prevent Sync from deleting files on an endpoint if a mount becomes unavailable. For instructions, see "Configuring Sync Endpoints" on page 243.

Using short-format option flags:

```
async
    -L "C:\Users\morgana\Aspera jobs\log"
```

```
    -N job1
    -i c:/users/morgana/.ssh/id_rsa
    -b "C:\Users\morgana\Aspera jobs\db"
    -l 500m
    -d c:/users/morgana/data
    -r morgana@10.0.0.1:/data
    -R /morgana/async/log
    -B /morgana/async/db
    -K bidi
    -t
```

Using long-format option flags:

```
async
    --alt-logdir="C:\Users\morgana\Aspera jobs\log"
    --name=job1
    --private-key-path=c:/users/morgana/.ssh/id_rsa
    --local-db-dir="C:\Users\morgana\Aspera jobs\db"
    --target-rate=500m
    --local-dir=c:/users/morgana/data
    --user=morgana
    --host=10.0.0.1
    --remote-dir=/data
    --remote-logdir=/morgana/async/log
    --remote-db-dir=/morgana/async/db
    --direction=bidi
    --preserve-time
```

If the session is between Linux computers, it also includes the following session options:

```
-u
-j
```

Or using long-format option flags:

```
--preserve-uid
--preserve-gid
```

async Command Reference

An **async** session accepts the following options, some of which are required.

Syntax

```
# async [instance_options] -N pair -d ldir -r [user@host:rdir] [session_options] ...
```

Note: Transfers started by `async` can be controlled from the HSTS GUI. Canceling an `async` transfer in the GUI shuts down `async`.

Required Command Options

Naming the async session: -N *pair*

`-N pair` is required in **async** commands. The value for *pair* is a name that uniquely identifies the Aspera Sync session and is visible in IBM Aspera Console. `-N pair` must follow any instance options and must precede all session arguments. Names can only use standard alphanumeric characters, plus "_" and "-" characters.

Note: If your remote host is an Aspera cluster, ensure that your session name is unique by naming the session with a descriptive string followed by the UUID of the local host, such as "cluster-sync-ba209999-0c6c-11d2-97cf-00c04f8eea45".

Specifying filepaths and filenames: *ldir* and *rdir*

ldir specifies the local directory to be synchronized and *rdir* specifies the remote directory to be synchronized. File paths and filenames must follow these rules:

- The drive letter is required in Windows paths, unless the server's `aspera.conf` file has a `docroot` defined for the user. If no drive letter is included when syncing with a Windows computer and `docroot` is not defined for the user, `async` displays the error message: "Failed. Peer error: Remote directory is not absolute."
- You can synchronize Windows, Linux, macOS, and other Unix-based endpoints and servers, but must take care with path separators. The path separator "/" is supported on Windows and other platforms. The path separator "\" is platform-agnostic *only* for the options `-d/r/L/R/B/b` and `--keep-dir-local/remote`. In Aspera Sync filtering rules, however, "\" is exclusively a quoting operator and "/" is the only path separator recognized.
- File names may not contain \n, \r, or \. Files with these in their names are skipped.
- When scanning or monitoring a file system for changes, `async` skips over files with names that end in one of the special suffixes specified in `aspera.conf` with `<resume_suffix>` and `<partial_file_suffix>` . To disable this behavior, you can set these values to the empty string. `<resume_suffix>` defaults to `.aspx`. The `<partial_file_suffix>` tag defaults to the empty string, but is often set to `.partial`.

Warning: If a source is on a NFS or CIFS mount, use `--local-mount-signature` or `--remote-mount-signature` (or both if both endpoints are on mounts and the Aspera Sync is bidirectional) to prevent Aspera Sync from deleting files on the non-mount endpoint if the mount becomes unavailable. For instructions on creating mount signature files, see "Configuring Sync Endpoints" on page 243.

Specifying the direction of the sync: -K *direction*

Aspera Sync has three modes of synchronization: `push`, `pull`, and `bidi`.

- `push`: The contents of *ldir* are synchronized to *rdir*, with the *ldir* content overwriting the *rdir* content, by default (unless the overwrite options are specified otherwise, such as to only overwrite if *rdir* is older, or never overwrite).
- `pull`: The contents of *rdir* are synchronized to *ldir*, with the *rdir* content overwriting the *ldir* content, by default.
- `bidi` (bi-directional): The contents of *ldir* and *rdir* are synchronized, with newer versions of files and directories overwriting older versions in either *ldir* or *rdir*, by default.

Using continuous mode: -C

Continuous mode is supported only when the file source is Windows, Linux, or macOS. See the following table for the operating system requirements for the Sync server and client for the different Syncdirections.

Continuous Sync Direction	Supported Sync Client OS	Supported Sync Server OS
PUSH	Linux, Windows, macOS	All
PULL	All	Linux, Windows, macOS
BIDI	Linux, Windows, macOS	Linux, Windows, macOS

One-time synchronization is supported between all operating systems.

The following tables are complete command-line options references. View an abbreviated version from the command line by running:

```
# async -h
```

For examples of **async** commands and output, see "Examples of Async Commands and Output " on page 267.

Environment Variables

If needed, you can set the following environment variables for use with `async`. 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_COOKIE=*cookie*
Set the transfer user cookie. Overridden by `--cookie`.

ASPERA_SCP_LICENSE=*license_string*
Set a base64-encoded Aspera license string.

ASPERA_SCP_PASS=*password*
Set the transfer user password. Overridden by `-w` and `--pass`.

ASPERA_SCP_TOKEN=*token*
Set the transfer user authorization token. Overridden by `-W` and `--token`.

Instance Options

`-A, --version`
Display the `async` version information and license information.

`--apply-local-docroot`
Prepend the local `docroot` to the local directory.

`-D[D..]`
Log at the specified debug level. Default is 0. Additional Ds return more messages.

`-h, --help`
Display help for command-line options.

`-L` *log_dir*, `--alt-logdir=`*log_dir*
Log to the specified logging directory on the local host. If the directory doesn't exist, `async` creates it for you.

`-q, --quiet`
Disable all output.

`--watchd=`*datastore*`:`*host*`:`*port*`[:`*domain*`]`
Use asperawatchd connected to the specified Redis for the transfer session. *datastore* can be `redis` or `scalekv`.

For example:

```
--watchd=redis:localhost:31415
```

The optional *domain* argument allows you to specify if the domain is other than the default root. For more information see “Using the Aspera Watch Service with Sync” on page 278.

Session Options

`-a` *policy*, `--rate-policy=`*policy*
Transfer with the specified rate policy. *policy* can be `fixed`, `fair`, `high`, or `low`. Default: `fair`

`--assume-no-mods`
Assume that the directory structure has not been modified. If a directory's modification time has not changed compared to the Aspera Sync database, **async** in non-continuous mode skips scanning the directory. This option makes scanning static directory structures faster. Aspera recommends using `--exclude-dirs-older-than` instead of this option.

`-B` *rdbdir*, `--remote-db-dir=`*rdbdir*
Save the remote database to the specified directory. Similar to `-b`, but applies to the remote database. For further usage information, see “The Sync Database” on page 249. Default: `.private-asp` at the root level of the synchronized directory. The directory is created if it does not already exist. If <async_db_dir> is set in `aspera.conf` on the server, that setting overrides the location specified with `-B .`

-b ***ldbdir*****, --local-db-dir=***ldbdir*****
Use the specified local database directory. Default: `.private-asp` at the root level of the synchronized directory.

You can save the Aspera Sync database to a different location than the default one under the *ldir* specified with `-d`. This allows you to store the database away from the main data files, which is useful for performance tuning. It is also useful when `-d` *ldir* is located on a network share volume that does not reliably support database locking. For more usage information, see "The Sync Database" on page 249.

-C, --continuous
Run continuous synchronization. Default: disabled.

Usage notes:

- Continuous mode is supported only when the file source is Windows or Linux. Continuous pulls can be run from any operating system if the source is Windows or Linux. Continuous push can be run only by Windows or Linux. Continuous bidi requires that both the Aspera Sync client and server are Windows or Linux.
- If a file is open, **async** cannot transfer the file due to sharing violations and might ignore the file if it is closed without changes. To specify the maximum number of retries after a sharing violation, use with `--sharing-retry-max`. To enable periodic scans that detect when an opened file has been closed and is ready for transfer, use with `--scan-interval`.
- If you receive an `inotify` error when attempting to run continuous synchronization, see "Troubleshooting Continuous Sync Errors" on page 287.

-c ***cipher*****, --cipher=***cipher***
Encrypt file data with encryption algorithm. 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 explicitly 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.

Value	Description	Support

`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.

- `NONE` - Do not encrypt data in transit. Aspera strongly recommends against using this setting.
- `AES128`, `AES192`, `AES256` - Use the GCM or CFB encryption mode, depending on the server configuration and version. Supported by all client and server versions.
- `AES128CFB`, `AES192CFB`, `AES256CFB` - Use the CFB encryption method. Supported by clients and servers version 3.9.0 and newer.
- `AES128GCM`, `AES192GCM`, `AES256GCM` - Use the GCM encryption mode. Supported by clients and servers version 3.9.0 and newer.

Default: `AES128`.

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 remote host key hash and fail on mismatch.

`--clean-excluded`
Remove excluded directories from `snap.db` on both Aspera Sync endpoints to decrease the size of `snap.db`. This option applies when directories are excluded by path (`--exclude`) or by modification time (`--exclude-dirs-older-than`). If the remote endpoint is running Aspera Sync older than 3.8.0, then the option is accepted (the session does not fail) but it has no effect on either endpoint.

`--compression={zlib|none}`
Compress a file before transfer using the specified method. Default: `none`.

--cookie=*cookie*
Specify a user-defined identification string to report to the Aspera Management interface. *cookie* cannot contain the special characters `\r`, `\n`, or `\0`.

--cooloff=*sec*
Delay the start of the transfer. For example, if `--cooloff=5`, `async` waits 5 seconds before copying a file. If `--cooloff=0` transfers start immediately. The client and server use the same cooloff period. Valid range for *sec*: integers 0-60. Default: 3.

--cooloff-max=*sec*
Wait up to the specified time (in seconds) for a file to stop changing before skipping synchronization of the file. Using this option prevents a one-time sync from waiting on a constantly changing file. The file is skipped and reported as an error. Default: 0 (disabled).

--create-dir
Create the source directory, target directory, or both if they do not exist, rather than reporting an error and quitting. Use with `-d` and `-r`.

-d *ldir*, --local-dir=*ldir*
Synchronize the specified local directory. Use `--create-dir` to create the remote directory if it does not already exist.

--dedup[=*mode*]
Take the specified the action when Aspera Sync detects duplicate files on the source, even if they have different pathnames. Requires `-k` with a full checksum. Available modes are `hardlink`, `inode` (only supported for Unix-based OSes), or `copy`. Default: `hardlink`.

- `hardlink` - When two or more source files are duplicates, a hardlink is created between them on the target. This saves storage by preventing multiple copies of the same file from accumulating on the target. The files on the target have the same inode, even if the source files have different inodes. The target must be running a Unix-based operating system. File metadata preservation options (`-u` and `-j`) are not supported with this option.
- `inode` - When two or more source files have matching inodes, a hardlink is created between them on the target and the target files have matching inodes. This option is supported only between Unix-based platforms. If `--dedup=inode` is used in a continuous sync, Aspera recommends using the `scan-interval` option.
- `copy` - After a file is synchronized on the target, the synchronized file is copied to the duplicate. This saves bandwidth by not transferring duplicate files. This mode is useful when the target is Windows. File metadata preservation options (`-u` and `-j`) are supported with this option.

Without the dedup option, all duplicate files are synchronized. Duplicates might still be synchronized, rather than hardlinked or copied, if one of the duplicates has not yet been synchronized on the target.

--delete-delay
Postpone the actual deletion of files or directories until the end of the synchronization. Use this option to prevent transfer delays that can occur when deletions are slow on the destination.

-E *file*, --exclude-from=*file*
Skip paths specified in the filter *file*. For more information on setting filters, see "Using Filters to Include and Exclude Files" on page 145.

--exclude="*pattern*"
Exclude paths that match *pattern*. Wildcards, such as * and ?, are supported but rules containing them must be in double quotes. For example, `--exclude="*.jpg"`. For more information, see "Using Filters to Include and Exclude Files" on page 145.

--exclude-dirs-older-than=*mtime*
After the initial scan, do not scan directories during subsequent synchronizations if they or their parents have a recursive modified time older than the specified value. The recursive modified time of a directory is the most recent modification time of it or any of its children (file or directory). Use this option to avoid rescanning directories that are known to be unchanged since the previous synchronization, such as a monthly archive directory structure in which only the most recent sub-directory is being modified.

mtime may be specified in any one of the following ways:

- As a positive number of seconds since 1970-01-01 00:00:00, for Unix and POSIX-compliant operating systems.

  Note: Some file servers, such as Windows NT, use a different epoch for the recursive modified time. In this case, *MTIME* should be specified as a duration relative to present or UTC timestamp.
- As a UTC timestamp with the format YYYY-MM-DDTHH:MM:SS, such as 2015-01-01T08:00:00.
- As a duration formatted as DDd HH:MM:SS or WWw DDd HHh MMm SSs. Directories whose "mtime" is older than Now minus *MTIME* are not scanned. **Input requirements:** Leading zero fields and spaces may be omitted. The leftmost fields are optional, but fields to the right of the largest unit specified are required. For example, to exclude directories older than 24 hours, you could specify 1d 0:0:0, 24:00:00, or 24h 00m 00s, but not 1d.

This option does not apply to the root directory.

Note: Aspera Sync stops and returns an error if the first run of **async** and the next run do not use the same `--exclude-dirs-older-than` option. If the first run specifies `--exclude-dirs-older-than`, then the next run must use this option, too. If the first run does not include `--exclude-dirs-older-than`, then the next run fails if this option is specified.

-G *size*, --write-block-size=*size*
Use the specified block size for writing. *size* is an integer with units of K, M, or bytes. Default: 64 MB.

-g *size*, --read-block-size=*size*
Set block size for reading. *size* is an integer with units of K, M, or bytes. Default: 64 MB.

-H *val*, --scan-intensity=*val*
Scan at the set intensity. *val* can be `vlow`, `low`, `medium`, `high`, or `vhigh`. `vlow` minimizes system activity. `vhigh` maximizes system activity by continuously scanning files without rest. Default: `medium`.

--host=*host*
Synchronize with the remote host that is specified by hostname or IP address. If the remote host is a cluster, enter the cluster DNS. When using `--host=`, the characters "@" and ":" are not treated specially in the argument to `-r` or `--remote-dir`. The transfer username cannot be specified as part of the remote directory filepath. Instead, it must be set with `--user=` or in the environment variable `$user` (on Windows, %USER%). Allowed forms are as follows:

```
--remote-dir user@host:/rdir # (old method)
--user user --remote-dir host:/rdir
--user user --host host --remote-dir /rdir
--remote-dir host:/rdir  # (uses $user)
--host host --remote-dir /rdir # (uses $user)
```

The following means the same as the first three lines above:

```
 -r /rdir --user=user --host=host
```

For backward compatibility, `-r A:/rdir` for any single letter A is still taken as a Windows path, not as `--host A -r /rdir`. To specify a one-letter host name *A*, use an explicit `--host=A`.

-I *file*, --include-from=*file*
Scan and include paths specified in the filter *file*. For more information, see “Using Filters to Include and Exclude Files” on page 145.

-i *file*, --private-key-path=*file*
Authenticate with the specified SSH private key file. For information on creating a key pair, see “Creating SSH Keys ” on page 152.

--ignore-delete
Do not copy removals to the peer. This option is used mostly with uni-directional syncs. In bi-directional sync, a deletion on one side is ignored but the next time **async** is run, the file is recopied from the other end. In continuous mode, the file is not recopied until either **async** is restarted or the file is changed (touched).

--ignore-mode
Do not synchronize file permissions of the source to the destination. This argument is useful when synchronizing from a Unix-like source to a Windows destination, which has different file permission behavior than the Unix-like source ("read only" files cannot be deleted or modified on Windows).

--ignore_remote_host_sync_name
Do not check that the remote host being used for the current transfer matches the host used when the local database was created.

--include="*pattern*"
Include paths that match *pattern*. Wildcards, such as * and ?, are supported but rules containing them must be in double quotes. For example, `--include="*.jpg"`. For more information on how to set include and exclude patterns, see "Using Filters to Include and Exclude Files" on page 145.

-j, --preserve-gid
Preserve file owner's *gid* when synchronizing files between Unix-like operating systems. Requires that **async** is running as root. Default: disabled.

-K *direction*, --direction=*direction*
Transfer in the specified direction. *direction* can be `push`, `pull`, or `bidi` (bi-directional). Default: `push`.

-k *type*, --checksum=*type*
Calculate the specified checksum type. *type* can be `sha1`, `md5`, `sha1-sparse`, `md5-sparse`, or `none`. A value of `none` is equivalent to a size check only and `async` will not detect a change in timestamp. Default: `sha1-sparse` for local storage, `none` for object storage.

--keep-dir-local=*dir*
Move deleted files into *dir*. The directory must exist (it is not created by `--create-dir`), and must be outside the synchronization directory (or excluded from the sync using `--exclude` or `--exclude-from`), but on the same file system.

--keep-dir-remote=*dir*
Move the server's deleted files into *dir*. The directory must exist (it is not created by `--create-dir`), and must be outside the synchronization directory (or excluded from the sync using `--exclude` or `--exclude-from`), but on the same file system.

-l *rate*, --target-rate=*rate*
Transfer no faster than the specified maximum transfer rate. *rate* is an integer with units of G/g, M/m, K/k, or bps. Default: 10 Mbps.

--local-force-stat
Force the local Aspera Sync to retrieve file information even if no changes were detected by scanning or file system notifications (equivalent to the behavior of Aspera Sync versions 3.8.1 and older). This option incurs a performance cost at the expense of immediately detecting file changes. See also `--remote-force-stat`.

--local-fs-threads=*number*
Use up to the specified number of threads to do file system operations on the local computer. Default: 1. This option is particularly useful when the local Sync directory is in cloud storage or mounted storage (NFS) where file system operations are slow. To set multiple threads for file system operations on the remote computer, use `--remote-fs-threads`.

--local-mount-signature=*signature file*
Verify that the local file system is mounted by the existence of this file. This option increases the time required to synchronize deletes. See also `--remote-mount-signature`.

-m *rate*, --min-rate=*rate*
Attempt to transfer no slower than the specified minimum transfer rate. *rate* is an integer with units of G/g, M/m, K/k, or bps. Default: 200 Kbps.

--mirror
For use only with **direction=push** or **direction=pull**. When the remote site is scanned:

- any destination file found missing on the remote site is sent from the source.

- any destination file that does not match the size or checksum of the source file is replaced by the file from the source.
- any destination file or directory that does not exist on the source is deleted

Note: This option is not compatible with: **--direction=bidi**, **--no-scan**, **--remove-after-transfer**, **--exclude-dirs-older-than**, and **--ignore-delete**.

-N ***pair*****, --name=*****pair***
Assign a name for the synchronization session. The value can contain only ASCII alphanumeric, hyphen, and underscore characters. This value is stored in the session cookie and can be used in IBM Aspera Console to identify the transfer session.

Note: `-N` must precede all session options.

-n ***action*****, --symbolic-links=*****action***
Handle symbolic links with the specified method, as allowed by the server. For more information on symbolic link handling, see “Symbolic Link Handling” on page 150.

action can be:

`copy` - create or update the link at the destination (default). Not valid for Windows source or destination.

`skip` - ignore the link altogether.

`follow` - treat the link as if it were the file or directory it points to, so that at the destination, what was a link is now a copy of the file or directory. Functions as `skip` if source is Windows.

--no-log=stats
Suppresses the STATS and PROG log messages.

--no-preserve-root-attrs
Disable the preservation of attributes on the Aspera Sync root.

--no-scan
Never scan. Use this option in a continuous **async** session to synchronize only new files (files that are added to the directory after the start of the **async** session) but not existing files. With `--no-scan`, Aspera Sync relies entirely on file system notifications to detect changes. As a result, if a directory is renamed after the **async** session starts, then the directory name is synchronized but the contents are not (because Aspera Sync does not recognize that the files were "moved" to the renamed directory). This option cannot be used with `--scan-interval` or one-time **async** sessions.

-O ***port*****, --udp-port=*****port***
Use the specified UDP port for FASP data transfer. Default: 33001.

-o ***policy*****, --overwrite=*****policy***
Overwrite files according to the specified policy, which can be `always`, `older`, or `conflict`. Use with `-K push` and `pull`. Default: `always` for `-K push` and `pull`; `conflict` for `-K bidi`.

Note: When syncing with object storage, only file size (`--checksum=none`) can be used to compare files. Thus, using `--overwrite=always` only overwrites files whose sizes have changed. If the content of a local file is different from a file with the same name in object storage but the files are the same size, the file in object storage is not overwritten. To overwrite files in this case, use `--overwrite=older`.

`--overwrite=older` is only accurate if the user also specifies `--preserve-time` (preserve timestamps).

To resolve `conflict` and `error` situations in a uni-directional sync, “touch” the problem files on the source and run **async** with `--overwrite=always`. This clears all `conflict` and `error` states as the problem files are synchronized.

-P ***port*****, --tcp-port=*****port***
Use the specified TCP port for SSH. *port* must be a valid numeric IP port. Default: 22.

--pending-max=***N***
Allow the maximum number of files that are pending transfer to be no more than the specified number. This option acts as a buffer. Default: 2000.

--preserve-access-time
Preserve file access time from the source to the destination. Default: disabled.

--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**.

- `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 - (Default) Do not preserve attributes. This mode is supported on all file systems.

Important Usage Information:

- This feature is only meaningful if both hosts are in a common security domain. If a SID (security ID) in a source file does not exist at a destination, the synchronization proceeds but no ACL data is saved and the log records that the ACL could not be applied.
- Both **--preserve-acls** and **--remote-preserve-acls** must be specified in order for the target side of a pull to apply the ACLs.
- ACLs are not synchronized when only the ACL is modified, or when only the ACL and filename are modified. ACLs are not preserved for directories.
- On Windows, the ACLs that are created for files that are transferred into user directories might restrict file access to specific users. Ensure that the ACLs on the transfer-cache directory (*`destination_path`*/*`session_name`*) are generic enough to allow access to all users who require it. For more information about the transfer-cache directory, see "The Sync Database" on page 249.

--preserve-creation-time
Preserve file creation time from the source to the destination. Valid only on Windows computers. Default: disabled.

--preserve-modification-time
Preserve file modification time from the source to the destination. Default: disabled.

--preserve-time
Preserve file timestamps. This is equivalent to `--preserve-modification-time` for Unix-based operating systems, and to `--preserve-modification-time --preserve-creation-time` on Windows. Default: disabled. Same as `-t`.

--preserve-xattrs={native|metafile|none}
Preserve extended file attributes data (xattr). 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 - (Default) Do not preserve attributes. This mode is supported on all file systems.

Important Usage Information:

- Xattr are not preserved for directories.
- If Aspera Sync is run by a regular user, only user-level attributes are preserved. If run as superuser, all attributes are preserved.

--proxy ***proxy_url***
Synchronize using the specified IBM Aspera Proxy address. The Proxy URL is specified with the following syntax:

`dnat[s]://proxy_username:proxy_password@proxy_ip_address[:port]`

The default port for DNAT is 9091, and for DNATS is 9092. The Proxy password must be specified or the synchronization fails to connect to the Proxy server.

-R ***rem_log_dir*****,--remote-logdir=*****rem_log_dir***
Use the specified logging directory on the remote host. The directory is created if it does not already exist. If <async_log_dir> is set in aspera.conf on the server, **async** initially logs to *rem_log_dir* but is then redirected to the directory specified for <async_log_dir>.

Note: -R cannot be used if the transfer user is restricted to aspshell.

-r ***rdir*****, --remote-dir=*****rdir***
Synchronize the specified directory on the remote host. *rdir* is [[*user*@]*host*:]*path*. If the target is the remote directory, you can use--create-dir to create the remote directory if it does not already exist.

CAUTION: If the source and target directories are both on the local host, do not specify a target directory that is inside your source directory.

--remote-force-stat
Force the remote Aspera Sync to retrieve file information even if no changes were detected by scanning or file system notifications (equivalent to the behavior of Aspera Sync versions 3.8.1 and older). This option incurs a performance cost at the expense of immediately detecting file changes. See also --local-force-stat.

--remote-fs-threads=***number***
Use up to the specified number of threads to do file system operations on the remote computer. Default: 1. This option is particularly useful when the remote Sync directory is in cloud storage or mounted storage (NFS) where file system operations are slow. To set multiple threads for file system operations on the local computer, use --local-fs-threads.

--remote-mount-signature=***signature file***
Verify that the remote file system is mounted by the existence of this file. This option increases the time required to synchronize deletes.

--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 the pull to apply the ACLs.

--remote-preserve-xattrs={native|metafile|none}
Like --preserve-xattrs but used attributes are stored in a different format on the remote computer. Defaults to the value of --preserve-xattrs.

--remote-scan-interval=***duration***
Set the scanning interval of the remote computer. See also --scan-interval.

--remote-scan-threads=***N***
Use the specified number of directory scanning threads on the remote computer. More threads decrease the time it takes for **async** to scan the directory after the initial synchronization, and increase the number of pending files. Default: 1. To specify the number of scanning threads on the local computer, see --scan-threads.

--remove-after-transfer
Remove source files after they are successfully synchronized.

--scan-dir-rename
Enable the detection of renamed directories and files compared to the previous scan, based on matching inodes. Enable the detection of renamed directories and files compared to the previous scan, based on matching inodes. When a new directory is found on the source and its inode matches

that of a previously found directory, it is considered a "rename" and the target directory is renamed accordingly. The source directory is scanned for content changes, and the target directory is updated accordingly.

Usage note:

- This option can be used only on file systems with persistent inodes, and does not work if inodes are volatile, as is the case with many network-mounted file systems.

--scan-file-rename
Enable the detection of renamed files compared to the previous scan, based on matching inodes. If a new file is found and its inode and last-modified time matches that of a previously found file that does not have multiple hardlinks, it is considered a "rename" and the remote file is renamed accordingly.

Usage note:

- This option can be used only on file systems with persistent inodes, and does not work if inodes are volatile, as is the case with many network-mounted file systems.
- If `--scan-file-rename` is used without `--scan-dir-rename`, a directory rename causes **async** to create a new directory and rename its files individually.

--scan-interval=*duration*
Enable periodic scans during a continuous Aspera Sync (a session run with the `-C` option) on the local host. *duration* is the interval between periodic scans and can be specified as DDd HH:MM:SS.mmm or WWw DDd HHh MMm SSs XXms XXus. Leading zero fields can be omitted. Spaces can be omitted. A plain number XX is interpreted as SSs (seconds).

--scan-threads=*N*
Use the specified number of directory scanning threads on the local computer. More threads decrease the time it takes for **async** to scan the directory after the initial synchronization, and increase the number of pending files. Default: 1. To specify the number of scanning threads on the remote computer, see `--remote-scan-threads`.

--sharing-retry-max=*N*
Retry synchronizations up to the specified maximum number after a sharing violation. The interval between retries is the number of seconds specified by `--cooloff`. Default: 3.

--symbolic-links=*action*
See `-n`.

-t
Preserve file timestamps. Same as `--preserve-time`.

--tags=*string*
User-defined metadata tags in JSON format that can be used in transfer session reporting and searches.

--tags64=*string*
User-defined metadata tags in JSON format and base64-encoded that can be used in transfer session reporting and searches.

--transfer-threads=*N[:size]*
Use the specified number of dedicated transfer threads and optionally specify the file size at which files are assigned groups of threads. The number of threads should not exceed the number of available CPU cores (the lower value of the client and server computers). If no size is specified, infinity is used as an upper bound.

For example, to use two transfer threads to transfer files smaller than or equal to 128 bytes and six transfer threads for all other files (those larger than 128 bytes), use the following options:

```
--transfer-threads=2:128 --transfer-threads=6
```

-u, --preserve-uid
Preserve the file owner's *uid* when synchronizing files between Unix-like operating systems. **async** must be run as root to use this option. Default: disabled.

--user=*user*
Authenticate the transfer with the specified username. With this option, the characters "@" and ":" are not treated specially in the argument to `-r` or `--remote-dir`.

-W *token_string*, --token=*token_string*
Use the specified authorization token. The token type (sync-push, sync-pull, or sync-bidi) must match the direction (push, pull, or bidi) of the requested transfer. The token path must match the remote directory of the requested transfer. If an invalid token is provided, the requested transfer will be denied.

-w *pass*, --pass=*pass*
Authenticate the transfer with the specified password.

--write-uid=*uid*
--write-gid=*gid*
Write files as the user *uid* or the group *gid*. *uid* and *gid* can be numeric, or by name. If by name, the name is looked up on the host performing the write. Failure to set the *uid* or *gid* is logged, but is not an error. The *uid* or *gid* is set after `ascp` completes and before moving the file from the staging directory to the final location.

`--write-uid` conflicts with `--preserve-uid`, and `--write-gid` conflicts with `--preserve-gid`.

-X *size*, --rexmsg-size=*size*
Use the specified *size* (in bytes) for a retransmission request. Maximum: 1440.

-x, --reset
Clear the Aspera Sync database and rescan the synchronized directories and files to create a fresh database. Default: off.

-Z *mtu*, --datagram-size=*mtu*
Use the specified datagram size. Value is an integer. Default: detected-path MTU.

Examples of Async Commands and Output

Examples of common Sync use cases and a description of **async** output.

Async Command Examples

1. **Continuous synchronization of a daily archive of large files on a Windows computer to Linux computer, preserving Windows ACLs, run as an async pull on the Linux computer:**

```
$ async -L /sync/logs -N backup -d /sync/backup -r alligator@everglades.company.com:"C:\data
\" -i /.ssh/lion_private_key -K pull --remote-scan-interval=4h --preserve-acls=metafile --
remote-preserve-acles=metafile -C --exclude-dirs-older-than=1w0d0h0m0s
```

   Details:

   - Logs are stored on the Linux computer in the specified location.
   - The user, lion, authenticates with an SSH key using the `-i` option
   - Because the files in the backup are large, `remote-scan-interval` is used to scan the Windows computer every 4 hours, which forces an additional scan in case any notifications are missed.
   - In order to preserve Windows ACLs in the backup, both `preserve-acls=metafile` and `remote-preserve-acls=metafile` must be specified.
   - Since the archive directory creates a new directory for each day, use `exclude-dirs-older-than=1w0d0h0m0s` to avoid scanning directories that are no longer changing (older than a week).

2. **High performance push synchronization of many (10,000s) of small files (<10 KB) between Windows computers:**

```
> async -L c:/logs:200 -q -N small-files -c none --pending-max=10000 --preserve-acls=native
--transfer-threads=4 -R c:/logs:200 -d c:/data/ -r bobcat@192.168.4.24:"C:\data\" -K push -l
500m
```

   Details:

- Specifying the logging locations (`-L` and `-R`) is optional. Adding `:200` to the end of the log directory value allows the logs to reach 200 MB before being rotated.
- If the connection is secure, disabling encryption using `-c none` may boost performance.
- Increase the number of pending files from the default of 2000 using `--pending-max=10000`.
- The `--preserve-acls=native` option preserves Windows ACLs.
- Using more FASP threads to move the data can improve performance, set with `--transfer-threads=4`. The number of threads should not exceed the number of CPU cores (the lower value of the client and server computers).
- The user must enter the password at the prompt because it is not provided in the command. Aspera recommends using SSH keys for authentication, but this is not required.

3. **Non-continuous bidirectional synchronization of directories containing a mix of large and small files in which small files are synchronized using one thread and large files use another, run on a Linux computer to a macOS computer:**

```
$ async -L /sync/logs -q -N sync-2017-01-01 -images --user=gazelle@company.com --
host=10.4.25.10 -r Library/daimages -i /lion/.ssh/lion_private_key -R Library/sync/logs --
transfer-threads=2:100000 -K bidi
```

   Details:

   - Logs are saved in the specified locations on both computers.
   - The user authenticates with an SSH key using the `-i` option.
   - The user and host are specified as separate options, rather than as part of the destination folder, so that the username with an @ can be used (@ is reserved in an `-r` argument for specifying the host).
   - The async session uses two threads, one for files larger than 100 KB and one for files less than or equal to 100 KB, specified with the `--transfer-threads` option.

4. **Non-continuous push synchronization through reverse IBM Aspera Proxy:**

```
$ async -N pushproxy -images -r lion@10.0.0.1:/daimages --proxy=dnats://
gazelle:password@10.0.0.4 -K push
```

   Details:

   - The transfer username on the destination (10.0.0.1) is lion, the Proxy IP address is 10.0.0.4, and the Proxy username is gazelle.
   - The Proxy URL option must include the Proxy user's password.

Async Output Example

When **async** is run in interactive mode, the status of each file in the synchronized directory is displayed in a list similar to the following:

```
/file1                          SYNCHRONIZED
/file2                          SYNCHRONIZED(exs)
/file3                          SYNCHRONIZED(skp)
/file4                          SYNCHRONIZED(del)
```

The status may be one of the following options:

- `SYNCHRONIZED`: file transferred
- `SYNCHRONIZED(skp)`: file skipped
- `SYNCHRONIZED(del)`: file deleted
- `SYNCHRONIZED(ddp)`: dedup (duplicate files present)
- `SYNCHRONIZED(exs)`: file exists
- `SYNCHRONIZED(mov)`: file has changed (renamed, moved, or different attributes)

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. 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

Filters can be specified on the **ascp** command line and in `aspera.conf`. Ascp applies filtering rules that are set in `aspera.conf` *before* it applies rules on the command line.

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
[!...]	[!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

You can use this procedure to test your filtering rules.

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 a server. For example:

   ```
   # ascp /tmp/src my_user_name@my_demo.example.com:Upload/
   ```

   Where the user is "my_user_name", and the target is the `Upload` directory.

   At the prompt, enter my_user_name's password.
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' my_user_name@my_demo.example.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 `my_demo.example.com` to `/tmp/dest`:

```
# ascp rules aspera@my_demo.example.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
```

Filtering Examples

Filtering examples that demonstrate the effects of adding more filter rules to the command and show how to format a filter rule file.

About this task

Note: You can synchronize Windows, Linux, macOS, and other Unix-based endpoints and servers, but must take care with path separators. The path separator "/" is supported on Windows and other platforms. The path separator "\" is platform-agnostic *only* for the options `-d/r/L/R/B/b` and `--keep-dir-local/remote`. In Aspera Sync filtering rules, however, "\" is exclusively a quoting operator and "/" is the only path separator recognized.

Procedure

1. Include files under top-level directories `Raw` and `Jpg`. Exclude all others.

```
# async ... --include='/Raw/**' --include='/Jpg/**' --exclude='*' \
--exclude='.*' ...
```

2. Same as Example 1, except also include directories starting with ".", at any level.

```
# async ... --include='.*/' --include='/Raw/**' --include='/Jpg/**' \
--exclude='*' --exclude='.*' ...
```

3. Same as Example 2, except exclude regular files ending in "~" or ".thm".

```
# async ... --include='.*/' --exclude='.*~' --exclude='*~' \
--exclude='.*.thm' --exclude='*.thm' --include='/Raw/**' \
--include='/Jpg/**' --exclude='*' --exclude='.*' ...
```

4. Same as Example 3, except include only certain directories under Jpg.

```
# async ... --exclude='.*~' --exclude='*~' --exclude='.*.thm' \
--exclude='*.thm' --include='.*/' --include '/Raw/**' \
--include='/Jpg/Big/**' --include='/Jpg/Med/**' \
--exclude='*' --exclude='.*' ...
```

The long sequence in Example 4 can also be represented as a file:

```
# async ... --exclude-from=- <<EOF
# no regular files with ~ suffix, dot or otherwise:
    .*~
    *~
# similarly for ".thm" suffix files:
    .*.thm
    *.thm
# include directories starting with "."
    + .*/
# include everything else found under top-level Raw :
    + /Raw/**
# and under Big/ and Med/ in Jpg:
    + /Jpg/Big/**
    + /Jpg/Med/**
# but nothing else:
    *
    .*
EOF
```

Bidirectional Example

Bidirectional synchronization syntax is similar to push or pull **async** sessions, as show in the following example.

Note: You can synchronize Windows, Linux, macOS, and other Unix-based endpoints and servers, but must take care with path separators. The path separator "/" is supported on Windows and other platforms. The path separator "\" is platform-agnostic *only* for the options `-d/r/L/R/B/b` and `--keep-dir-local/remote`. In Aspera Sync filtering rules, however, "\" is exclusively a quoting operator and "/" is the only path separator recognized.

Example Options:

- Pair name = "asyncTwoWay"
- Local directory is /fio/S
- Remote directory and login is admin@192.168.200.218:d:/mnt/fio/S (Windows computer)
- Password is v00d00
- Target rate = 100,000 Kbps or 100 Mbps
- Transfer policy = fair
- Read-block size = 1048576 or 1MB
- Write-block size = 1048576 or 1MB
- Continuous transfer
- Bidirectional transfer

Example Command:

```
$ async -N asyncTwoWay -d /fio/S -r admin@192.168.200.218:d:/mnt/fio/S -w v00d00 -l 100M -a
fair -g 1M -G 1M -C -K BIDI
```

Example Output:

```
/                                                    SYNCHRONIZED
```

```
/a                                                              SYNCHRONIZED
/b                                                              SYNCHRONIZED
/c                                                              SYNCHRONIZED
/DIR1                                                           SYNCHRONIZED
/A1                                                             SYNCHRONIZED
/DIR2                                                           SYNCHRONIZED
/A2                                                             SYNCHRONIZED
/REMOTE_DIR1                                                    SYNCHRONIZED
/REMOTE_DIR2                                                    SYNCHRONIZED
/REMOTE_DIR1                                                    SYNCHRONIZED(del)
/DIR1/a                                                         SYNCHRONIZED
/DIR1/b                                                         SYNCHRONIZED
/DIR1/c                                                         SYNCHRONIZED
[idle    ] Found/synchronized/Pending/Error/Conflict=9/9/0/0/0
```

Synchronizing with AWS S3 Storage

Sync can be used to synchronize files when the source or destination is AWS S3 Cloud Object Storage. Each endpoint (HSTS) of the **async** session must be configured to support Sync and the **async** must include certain file system-related options.

About this task

Capabilities:

- Non-continuous PUSH, PULL, and BIDI synchronization between a local disk and AWS S3, as well as between S3 buckets.
- Continuous PULL and BIDI when S3 is the content source; requires the `--scan-interval` option.

Requirements:

- An IBM Aspera On Demand instance in AWS S3, or HSTS for Linux or Windows version 3.7.3 or later installed on a virtual machine instance in AWS with Trapd enabled. For instructions on setting up a HSTS in the cloud, see the High-Speed Transfer Server Admin Guide for Linux: Enabling AWS EC2/AWS S3 Using the Command Line.
- The S3 instance must have an On Demand entitlement and a Aspera Sync-enabled license.
- The **async** binary must be installed on both the source and destination server.
- Configure the S3 instance, or both S3 endpoints if you are running an S3-to-S3 synchronization, as described in the following steps.

Procedure

1. SSH into your instance as root by running the following command.

   The command is for Linux but also works for Mac. Windows users must use an SSH tool, such as PuTTy.

   ```
   # ssh -i identity_file -p 33001 ec2-user@ec2_host_ip
   ```

2. Elevate to root privileges by running the following command:

   ```
   #  su -
   ```

3. Set an S3 docroot for the system account user that will be used to run **async**.

   ```
   # asconfigurator -x "set_user_data;user_name,username;absolute,s3://s3.amazonaws.com/
   bucketname"
   ```

   If you are not using IAM roles, then you must also specify the S3 credentials in your docroot:

   ```
   s3://access_id:secret_key@s3.amazonaws.com/my_bucket
   ```

   By setting the docroot for the system user, the account becomes an Aspera transfer user.

4. Set database and log directories for **async**.

These directories must be located in `/mnt/ephemeral/data`. The `/mnt/ephemeral/` directory is no-cost ephemeral storage that is associated with your instance. Aspera recommends creating a directory to use that is named for the transfer user, and giving the transfer user write access. For example, if the transfer user is `ec2_user`, run the following commands to create the directory `/mnt/ephemeral/data/ec2_user`, create the database and log subdirectories, give `ec2_user` write access, and set the directories as the location for the database and logs:

```
# mkdir /mnt/ephemeral/data/ec2_user
# mkdir /mnt/ephemeral/data/ec2_user/db
# mkdir /mnt/ephemeral/data/ec2_user/log
# chown -R ec2_user /mnt/ephemeral/data/ec2_user
# asconfigurator -x "set_node_data;async_db_dir,/mnt/ephemeral/data/ec2_user/db"
# asconfigurator -x "set_node_data;async_log_dir,/mnt/ephemeral/data/ec2_user/log"
```

Examples of Sync to or from S3

About this task

Note: If the client is on the cloud storage host, the following options are required:

- The log directory and local database directory must be specified by using the `-L` and `-b` options.
- The `--apply-local-docroot` option must be used in order to transfer content into the object storage, rather than the local disk.

The following examples include the optional arguments `--transfer-threads`, `--local-fs-threads`, and `--remote-fs-threads`, which improve performance when one or both endpoints are in cloud storage.

One-time push from local disk to S3:

A one-time (non-continuous) push that is run from a local disk to an S3 bucket using SSH keys (for more information on using SSH keys, see “Creating SSH Keys ” on page 152), where `ec2_user` is the transfer user:

```
# async -N sync-to-s3 -d /data/data-2017-01 -r ec2_user@192.0.4.24:/data -i /bobcat/.ssh/
private_key -K push -B /mnt/ephemeral/data/db --transfer-threads=8 --remote-fs-threads=16
```

One-time bidi from S3 to local disk:

A one-time bidirectional sync that is run from the S3 client to a local disk:

```
# async -L /mnt/ephemeral/data/log --apply-local-docroot -N bidi_london -d /data -r
bear@192.0.12.442:/data -K bidi -b /mnt/ephemeral/data/db -B /async/log --transfer-threads=8 --
local-fs-threads=16
```

One-time pull from S3 to S3:

A one-time pull by `ec2_user` from `s3host` to `/data/2017` in the client S3 storage:

```
# async -L /mnt/ephemeral/data/log --apply-local-docroot -N s3sync -d /data/2017 -r
ec2_user@s3host:/data/2017-01 -K pull -b /tmp --transfer-threads=8 --local-fs-threads=16 --
remote-fs-threads=16
```

Sync with Basic Token Authorization

Aspera nodes that require access key authentication, such as IBM Aspera Transfer Cluster Manager or IBM Aspera on Cloud transfer service (AoCts), can be used as synchronization endpoints by configuring the **async** database on the node and authenticating the **async** session with a basic token. A basic token requires a docroot on the server and allows access to all files in the docroot.

About this task

Procedure

1. On the client, set a location for the Sync snapshot database by running the following command:

```
# asconfigurator -x "set_node_data;async_db_dir,filepath"
```

2. On the server, set a docroot for the transfer user.

   Log in or SSH into the server and run the following command:

```
# asconfigurator -x "set_user_data;user_name,username;absolute,filepath"
```

3. Create an Aspera access key.

   ATC Manager: Open the Cluster Manager web UI. Click your cluster and click the **Access Keys** tab. Click **New** and fill in the required information (for a description of the fields, see the Aspera Transfer Cluster Manager Admin and Usage Guide).

   AoCts: See https://ibm.ibmaspera.com/helpcenter/transfer-service/managing-access-keys/transfer-service-access-keys.

4. Create the basic token from the access key ID and secret.

   Run the following command to encode the access key ID and secret in base64.

```
# echo -n access_key_id:secret | base64
```

   The basic token looks similar to the following:

```
ZGlEZXVGTGNwRzlJWWRzdnhqMFNDcTRtT29oTkpUS3ZwNVEyblJXakRnSUE6YXNwZXJh
```

   If the basic token breaks across lines in the output, rerun the command using the `-w0` option to remove the line break. For example:

```
# echo -n diDeuFLcpG9IYdsvxj0SCq4mOohNJTKvp5Q2nRWjDgIA:aspera | base64 -w0
```

5. Run a synchronization, using the basic token.

   Run **async** with the `-W` option or set the ASPERA_SCP_TOKEN environment variable. For example,

```
# async -N atcm_sync -images -r lion@10.0.0.1:/daimages -K push -W "Basic
ZGlEZXVGTGNwRzlJWWRzdnhqMFNDcTRtT29oTkpUS3ZwNVEyblJXakRnSUE6YXNwZXJh"
```

   If you are synchronizing with an ATC Manager node, specify its location as its IP address, rather than the DNS name, to ensure that the file is transferred to the correct node.

Using the Aspera Watch Service with Sync

Sync can use asperawatchd for more efficient file system change detection, particularly for file systems with many files.

Starting Aspera Watch Services and Creating Watches

The Aspera Watch Service (**asperawatchd**) is a file system change detection and snapshot service that is optimized for speed, scale, and distributed sources. On file systems that have file system notifications, changes in source file systems (new files and directories, deleted items, and renames) are detected immediately, eliminating the need to scan the file system. On file systems without file notifications, such as object storage, Solaris, and AIX file system scans are automatically triggered.

About this task

The Aspera Watch Service can be used on any local or shared (CIFS, NFS) host. However, when watching mounted shared storage and the change originates from a remote server, the Watch Service does not receive file notifications. In such cases, set <scan_period> in `aspera.conf` to frequent scans, such as 1 minute. See the following steps for instructions.

When used in conjunction with **ascp** commands, the Aspera Watch Service enables fast detection and transfer of new and deleted items. For more information on using watches with **ascp**, see "Transferring and Deleting Files with the Aspera Watch Service" on page 238.

To start the Aspera Watch Service and subscribe to (create) a watch:

Procedure

1. Configure a docroot or file restriction for the user.

   Docroots and path restrictions limit the area of a file system or object storage to which the user has access. Users can create Watch Folders and Watch services on files or objects only within their docroot or restriction.

   Note: Users can have a docroot or restriction, but not both or Watch Folder creation fails.

   To set up a docroot from the command line, run the following command:

   ```
   # asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
   ```

   Restrictions must be set from the command line:

   ```
   # asconfigurator -x "set_user_data;user_name,username;file_restriction,|path"
   ```

   The restriction path format depends on the type of storage. In the following examples, the restriction allows access to the entire storage; specify a bucket or path to limit access.

Storage Type	Format Example
local storage	For Unix-like OS: • specific folder: `file:////folder/*` • drive root: `file:////*` For Windows OS: • specific folder: `file:///c%3A/folder/*` • drive root: `file:///c*`
Amazon S3 and IBM Cloud Object Storage - S3	`s3://*`
Azure	`azu://*`
Azure Files	`azure-files://*`
Azure Data Lake Storage	`adl://*`
Alibaba Cloud	`oss://*`
Google Cloud	`gs://*`
HDFS	`hdfs://*`

   With a docroot or restriction set up, the user is now an Aspera transfer user. Restart asperanoded to activate your change:

   Run the following commands to restart `asperanoded`:

   ```
   # systemctl restart asperanoded
   ```

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

   ```
   # service asperanoded restart
   ```

2. Ensure the user has permissions to write to the default log directory if no directory is specified.

   For more information about configuring log directories, see"Watch Service Configuration" on page 235.

3. Configure Watch Service settings.

   Though the default values are already optimized for most users, you can also configure the snapshot database, snapshot frequency, and logging. For instructions, see "Watch Service Configuration" on page 235.

4. Start a Watch Service under the user.

   The following command adds the Watch Service run under the user to the Aspera Run Service database:

   ```
   # /opt/aspera/sbin/asperawatchd --user username [options]
   ```

5. Verify that the Watch Service daemon is running under the user.

   Use the **aswatchadmin** utility to retrieve a list of running daemons. Daemons are named for the user who runs the service. For example, if you started a Watch Service under root, you should see the root daemon listed when you run the following command:

   ```
   # /opt/aspera/bin/aswatchadmin query-daemons
   [aswatchadmin query-daemons] Found a single daemon:
       root
   ```

6. Create a watch.

   A watch is a path that is watched by the Aspera Watch Service. To create a watch, users subscribe to a Watch Service and specify the path to watch. run the following command, where *daemon* is the username used to start the `asperawatchd` service and *filepath* is the directory to watch:

   ```
   # /opt/aspera/bin/aswatchadmin subscribe daemon filepath
   ```

   When you create a new subscription, you can also set watch-specific logging, database, scan period, and expiration period, and override `aspera.conf` settings.

   Note: The default scan period is 30 minutes. If you are watching a file system that does not support file system notifications (such as object storage, mounted storage (NFS), Solaris, and AIX), Aspera recommends setting a more frequent scan to detect file system changes quicker.

   For more information on using these options, see "Managing Watch Subscriptions" on page 237 or run:

   ```
   # /opt/aspera/bin/aswatchadmin subscribe -h
   ```

   Note: The default expiration for watches is 24 hours. If a watch subscription expires before the user resubscribes to it, a new subscription must be created.

Starting the Aspera Watch Service

Sync can be configured to use asperawatchd for fast synchronization of very large numbers of files without scanning the directory. Synccan push files from a local directory, pull files from a remote directory, or create a bi-directional session between two directories (as long as asperawatchd is properly configured to monitor both directories).

About this task

To **push** files to a remote server using Aspera Watch Service, configure **asperawatchd** on the local host. The remote server does not need to be configured. For instructions on starting asperawatchd for a push Sync, see "Starting Aspera Watch Services and Creating Watches" on page 233.

To **pull** files from a remote host, configure **asperawatchd** on the remote host. See the following steps for configuration instructions. The local host does not need to be configured.

Procedure

1. On the remote server, configure asperawatchd database storage.

If you set default database storage, Sync uses asperawatchd for all pull requests to the server, whereas if you set database storage for a specific user then asperawatchd is used only for pull requests by that user.

To configure the Watch Service database as the default, run the following command:

```
#asconfigurator -x "set_node_data;async_watchd,redis:hostname:31415[:domain]"
```

To configure the database storage for a specific user, run the following command:

```
#asconfigurator -x
"set_user_data;user_name,username;async_watchd,redis:hostname:31415[:domain]"
```

2. On the remote server, set up asperawatchd.

   For instructions, see “Starting Aspera Watch Services and Creating Watches” on page 233.

Watch Service Configuration

The Aspera Watch Service configuration in the `<server>` section of `aspera.conf` includes the snapshot database, snapshot frequency, and logging:

```
<server>
    <rund>...</rund>
    <watch>
        <log_level>log</log_level>
        <log_directory>AS_NULL</log_directory>
        <db_spec>redis:host:31415:domain</db_spec>
        <watchd>
            <max_directories>1000000</max_directories>
            <max_snapshots>10000</max_snapshots>
            <snapshot_min_interval>3s</snapshot_min_interval>
            <snapshot_min_changes>100</snapshot_min_changes>
            <scan_threads>16</scan_threads>
        </watchd>
        <watchfolderd>...</watchfolderd>
    </watch>
</server>
```

To view current settings without opening `aspera.conf`, run the following command and look for settings that start with `watch` and `watchd`:

```
# /opt/aspera/bin/asuserdata -a
```

Note: Logging and database settings apply to both the Watch Service and Watch Folders services.

Configuring Watch Service Settings

Configure the Watch Service by using **`asconfigurator`** commands with this general syntax:

```
# /opt/aspera/bin/asconfigurator -x "set_server_data;option,value"
```

Options and values are described in the following table.

Configuration Options and Values

asconfigurator option **`aspera.conf` setting**	**Description**	**Default**
watch_log_dir <log_dir>	Log to the specified directory. This setting applies to both the Watch Service and Watch Folders services.	The Aspera logging file (“Log Files” on page 356).

asconfigurator option aspera.conf setting	Description	Default
watch_log_level <log_level>	The level of detail for Aspera Watch Service logging. This setting applies to both the Watch Service and Watch Folders services. Valid values are `log`, `dbg1`, and `dbg2`.	`log`
watch_db_spec <db_spec>	Use the specified Redis database, which is defined with the syntax `redis:`*`ip_address`*`:`*`port`*`[:`*`domain`*`]`. This setting applies to both the Watch Service and Watch Folders services.	`redis:127.0.0.1:31415`
watchd_max_directories <max_directories>	The maximum number of directories that can be watched (combined across all watches). This setting is used only on Linux machines to overwrite the system value `/proc/sys/fs/inotify/max_user_watches`. To overwrite the system value with the `aspera.conf` value, run the setup procedure in the admin tool: `# aswatchadmin setup`	1000000
watchd_max_snapshots <max_snapshots>	The number of snapshots that are stored in the database before the oldest are overwritten.	10000
watchd_snapshot_min_interval <snapshot_min_interval>	The maximum amount of time between snapshots. If this period passes without the minimum number of changes to trigger a snapshot, a new snapshot is taken.	3s
watchd_snapshot_min_changes <snapshot_min_changes>	The minimum number of changes that trigger a snapshot. If this number is reached before the snapshot minimum interval passes, a new snapshot is taken.	100
watchd_scan_threads <scan_threads>	The number of threads to use to scan the watched folder. More threads increase the speed of the scan, particularly for folders with large numbers of files, but require more of your computer's resources.	16

Sync with Aspera Watch Service Session Examples

Examples of **async** commands for push, pull, and bidi sessions that use asperawatchd to identify files to transfer.

Push Example

Configure and start asperawatchd on the local host to push files with asperawatchd (see “Starting Aspera Watch Services and Creating Watches” on page 233).

To push files, start a Sync session with the `--watchd` *`datastore:host:port[:domain]`* option. For example:

```
async --watchd redis:localhost:31415:root -N watch_push -d /data/D1 -r adminuser@10.0.0.1:/
data/R1
```

Pull Example

Configure and start asperawatchd on the remote host to pull files with asperawatchd (see “Starting Aspera Watch Services and Creating Watches” on page 233).

Sync reads the remote host's `aspera.conf` file to determine whether or not to use asperawatchd for the session. To pull files, start a Sync session with the `-K pull` option. For example:

```
async -N watch_pull -d /data/D1 -r adminuser@10.0.0.1:/data/R11 -K pull
```

Bidirectional Example

Configure and start asperawatchd on the local and remote hosts to start a bidirectional session with asperawatchd (see “Starting Aspera Watch Services and Creating Watches” on page 233).

To synchronize bidirectionally, start a Sync session with the `--watchd` *`datastore:host:port:domain`* option and the `-K BIDI` option. For example:

```
async --watchd redis:localhost:31415:root -N watch_session -d /data/D1 -r adminuser@10.0.0.1:/
data/R11 -K BIDI
```

Remote from ascp Example

If you are using CIFS or NFS mounted storage, you must configure and run asperawatchd service on the host running the NFS server, but neither the local host nor the remote host need to be configured.

On the NFS server, you must also set the Redis database to a non-loopback address by configuring Redis with a modified configuration file including the correct port and host address bindings. For example, if your host address is "10.54.44.194":

```
# Accept connections on the specified port, default is 6379.
# If port 0 is specified Redis will not listen on a TCP socket.
port 31415

# If you want you can bind a single interface, if the bind option is not
# specified all the interfaces will listen for incoming connections.
#
bind 10.54.44.194
```

Save your configuration file and then run the asperaredisd service with the location of your configuration file.

```
# /opt/aspera/sbin/asperaredisd /filepath/redis_configuration.conf.
```

Point asperawatchd to the new Redis location by running the following command on your server:

```
# asconfigurator -x "set_node_data;watchfolderd_db_spec,redis:redis_host:redis_port:domain"
```

For example,

```
# asconfigurator -x "set_node_data;watchfolderd_db_spec,redis:10.54.44.194:31415:root"
```

Restart asperawatchd.

```
# /opt/aspera/bin/asperawatchd --user username
```

You can now start a Sync session from any client mounting NFS storage from that NFS server.

Important: The path of your mounted directory must match the path of the directory on the NFS server. For example, if the directory is found at /data/D1 on the NFS server, you must mount it at /data/D1.

Start a Sync session with the local directory (`-d`) pointing to the mounted storage and the `--watchd` option pointing to the remote Redis monitored by asperawatchd. For example:

```
async --watchd redis:10.54.44.194:31415 -N watch_remote -d /data/D1 -r adminuser@10.0.0.1:/
data/R11 -K BIDI
```

In this example, the client on Host A starts the Sync session. The asperawatchd service on Host B (10.54.44.194) scans the `/data/D1` directory mounted by Host A and passes the snapshot to Sync. Synctransfers the relevant files from the mounted storage to the target directory remote Host C (10.0.0.1). In this example, only Host B needs to be configured for asperawatchd.

Note: These examples are all one-time sessions, but you can run any of these sessions in continuous mode (if the source machine is Windows or Linux) by using the `-C` option. In continuous mode, any changes you make to a monitored directory are detected by "Introduction to Watch Folders and the Aspera Watch Service" on page 183. Changes are propagated through Aspera Sync.

Sync Monitoring and Logging

Admins can use the **asyncadmin** command-line tool to monitor **async** sessions and snapshots. Sync logs offer detailed information about session events, such as transfers and conflicts.

asyncadmin Command-Line Options

Administrators can use the **asyncadmin** tool to view the status of the current synchronization, as well as the latest snapshot. This includes the number of files in each state and any changes that might be incomplete on the remote endpoint. **asyncadmin** also offers troubleshooting options for deleting file records from a snapshot by path globbing match or filename. Learn more about **asyncadmin** definitions, allowable values, and defaults.

General **asyncadmin** usage:

```
# asyncadmin -d path [-N name][options]
```

The `-N` *name* option is required if multiple Syncsessions are running; you must specify the name of the session to which the **asyncadmin** command should apply.

Note: When records are deleted using the `-M` or `-E` options, Syncrecalculates file counters for the entire database. This can take a while, depending on the size of the database.

Session Options

`-A`
Display the **asyncadmin** version.

`-b` *`path`*`, --local-db-dir=`*`path`
Specify the local database directory. The default location is the local Syncdirectory.

`-C, --clean`
Delete problem records (records with statuses of CONF, PCONF, and ERR).

`-d` *`path`*`, --local-dir=`*`path`
Specify the local Sync directory.

`-E` *`number`*`, --erase=`*`number`
Delete the specified file record by number.

`-F, --force`
Allow changes while database is in use.

`-f, --file-info`
Report the status of all files.

CAUTION: The use of this option is not recommended on Windows, as it can cause the database to lock and **async** to fail. An alternative is to use the **`-t`** option.

-h, --help
Display the **asyncadmin** command-line option help.

-j, --journal
List the changes that might be incomplete remotely.

-l, --list
List the snapshot databases found in the database directory.

-M *pattern*, --match=*pattern*
Delete file records that have paths that match the specified pattern (path globbing).

-m, --meta
Report metadata.

-N *name*, --name=*name*
Select a source-destination pair from the snapshot database by name.

-O, --compact
Compact the database file.

-p, --pause
Pause when displaying a large amount of file data (for example, **-f**).

-q, --quiet
Display only the requested information. Use with **-f** / **--file-info** to disable abbreviating file names in the output.

-s, --summary
Report the number of files in each state. When **-s** is used alone, a brief summary from the **async** database's counters table is reported back (same as the cached counters as in the **-t** option).

CAUTION: The use of this option is not recommended on Windows, as it can cause the database to lock and **async** to fail. An alternative is to use the **-t** option below.

-s -v
When **-s** is used with **-v**, every record in the **async** database is counted.

Important: This should only be used when **async** is not running.

-T, --terminate
Shut down **async** if it is running. This option cannot be used if the storage style set for <async_db_spec> is LMS and outputs an error message.

-t *num*, --tail=*num*
Report status of last *num* files.

Note: Use of this option on Windows as an alternative to the **-f** and **-s** options above.

--touch=*path*
Change the recursive mtime of the node and all its parents to current time if they are older. This option is only applied if **async** has been run using the **--exclude-dirs-older-than** option.

-v, --verbose
Increase the verbosity of summary (**-s**) or file info (**-f**).

-x, --init
Delete all file system snapshot records.

Logging

By default, Sync logs all file system synchronization events and transfers, including any errors that were encountered by synchronizing hosts, to `syslog`. You can set the logging location on both endpoints when you start **async**.

Important: If you attempt to synchronize a directory without the proper read/write permissions, the directory and files it contains are *not* marked with an ERROR flag in the file directory status output. However, the error will be noted in the log file.

Troubleshooting Sync

Many Sync problems can be corrected by using required options, ensuring users have necessary permissions to access files, and configuring the endpoints as required.

Troubleshooting General Sync Errors

Fixes for common Sync issues.

The Sync client displays failure to start sync error

When the **async** binary on the remote computer cannot initialize, the **async** client gets a generic error similar to the following:

```
Failed to start sync session
```

Causes: Possible causes include the following:

- **async** binary doesn't exist (or is not in the path and sshd cannot find/execute it).
- **async** binary cannot be run.
- **async** binary cannot initialize properly (such as when the system is out of memory or socket resources).
- **async** binary cannot create its log files, if specified with `-R` (bad path, bad permissions).

Solutions:

- Confirm that the **async** binary is present. Look in the following location:

  ```
  /opt/aspera/bin/
  ```

- Confirm that the Aspera license shows that Sync is enabled. Run the following command and look for `sync2` in the list of enabled settings:

  ```
  # ascp -A
  ```

- If the system is under-resourced, increase the timeout allowed between the start of an **async** session and the FASP transfers associated with it by running the following command. In this example, the timeout is increased to 10 minutes (600 seconds):

  ```
  # /opt/aspera/bin/asconfigurator -x "set_node_data;async_connection_timeout_sec,600"
  ```

- Confirm that the path in the argument for `-R` is valid and that the Syncuser has write permissions to the directory.

Never-ending bidirectional session, with one file stuck in "pending" state

Causes: This can happen if a file is not in error for Sync but is in error for the underlying **ascp** process. For example, when **async** is run with `--checksum=none` and access to the file is denied, **async** does not open the file to calculate a checksum so it does not recognize that the file is unavailable, but **ascp** cannot open the file and reports an error. This can also happen if a file is truncated during the initial synchronization; the server **ascp** reports an error but the client **ascp** does not.

How to recover: Stop the Sync session by running the following command:

```
# asyncadmin -d path -N name -T
```

Check file permissions on the source and destination, and confirm that files are no longer being modified. Rerun your Sync session. You do not need to use `-x`.

Async fails with no specific reason

Causes: This can happen if the **async** user does not have permission to the files. This problem often arises when scripts are used to write files to one of the endpoints and the system permissions are overridden. Check the user's permissions to the files.

How to recover: Stop the **async** session by running the following command:

```
# asyncadmin -d path -N name -T
```

Edit the script to write files with the correct permissions, and rerun the **async** session.

Error returned when you try a synchronization from Linux to Windows.

When you try to synchronize from Linux to Windows, you receive the following error:

```
Failed. Peer error: Symlink policy copy not supported on Windows peer.
```

Solution: Specify `-n skip` or `--symbolic-links=skip` when performing the synchronization.

Error returned when you synchronize two locations on the same computer

You can synchronize files between two locations on the same computer. If you only enter the "remote" directory, such as `-r /tmp/`, then **async** fails with the following error:

```
Failed - Error, must specify remote-host name
```

Solution: Specify the remote host and path as `-r username@127.0.0.1:filename`.

Troubleshooting Continuous Sync Errors

In continuous mode, Sync can encounter operating system-related issues. The following article describes how to fix several of these.

Error returned when you attempt a continuous synchronization

If you attempt to run a continuous Sync from a client that does not support continuous mode, you receive the following error:

```
Failed. File system change notification not supported by platform (code=45112)
```

If you attempt to run a continuous Sync to a server that does not support continuous mode, you receive the following error:

```
Failed. [PEER} File system change notification not supported by platform (code=45112)
```

Solution: You must run your Sync session to or from a computer with an operating system that supports continuous mode:

Continuous Sync Direction	Supported Sync Client OS	Supported Sync Server OS
PUSH	Linux, Windows, macOS	All
PULL	All	Linux, Windows, macOS
BIDI	Linux, Windows, macOS	Linux, Windows, macOS

If that is not possible, you have two options for a workaround:

1. You can run **async** as a cron job that detects file system changes with asperawatchd. For more information, see “Starting Aspera Watch Services and Creating Watches” on page 233.

2. You can run **async** in continuous mode on source systems whose operating systems do not support file notifications by using `--scan-interval`. This enables periodic scanning of the file system to detect changes. The periodic scan is less efficient than the Aspera Watch Service file system monitoring.

Error returned when you attempt to monitor a Linux directory in continuous mode

If you attempt a continuous **async** session and the source is a Linux computer, you might receive the following error:

```
Failed to set up directory change notification - reached the per-user limit on number of
inotify watch descriptors.
```

Cause: You have exceeded the per-user limit imposed by the OS on the number of directories that can be monitored (determined by the number of inotify watch descriptors).

How to recover: You must modify the kernel parameters on the Linux computer to increase the maximum number of user watches. The following procedure might differ between Linux versions; consult your operating system Administrator's guide for more information.

1. On the Linux computer, open `/etc/sysctl.conf` in a text editor and increase the maximum number of user watches. Enter a value that exceeds the maximum number of directories ever expected to exist in the docroot that is monitored by **async**. For example,

   ```
   fs.inotify.max_user_watches=1000000
   ```

2. Save your changes.
3. Load the configuration changes by running the following command:

   ```
   # sysctl -p
   ```

4. Confirm that the changes took effect by running the following command:

   ```
   # sysctl -a | grep max_user_watches
   fs.inotify.max_user_watches=1000000
   ```

Resolving Bidirectional Sync File Conflicts

When run in bidirectional mode, Sync reports file conflicts when a file was modified on both endpoints and Sync cannot determine which version to use.

For example, you have computer A and computer B and you want to synchronize the following directory and files on both computers:

```
My_documents
---Document1
---Document2
---Document3
```

If `Document2` is changed on both computer A and computer B, then when you run the **async** session, Sync reports the conflict:

```
async -N my_bidi_sync -d /my_documents -r colleague@B:/home/my_documents -w pass -K bidi
/              SYNCHRONIZED
/Document1     SYNCHRONIZED
/Document2     CONFLICT
/Document3     SYNCHRONIZED
```

Both versions of `Document2` are left intact and you must manually resolve the conflict between them.

Resolve the conflict using one of the following methods, depending on if you have access to both endpoints (use method 1 or 2), which changes you want to preserve, and how soon you need resolution:

1. Reconcile the files

The slowest method, but it preserves changes and resolves the issue immediately (once files are edited).

If you have access to the file on both endpoints, compare the files and edit them until they are no longer different. To use a utility like `diff`, use **ascp** or other means to copy the remote file onto your local computer in a different directory from the local conflicted file.

Verify that the two files are no longer conflicted by comparing their checksums. Run the following command for each file to calculate its checksum:

```
# md5sum filepath
```

If the checksums match, then you can run the **async** session again and the files are synchronized without conflict.

```
async -N my_bidi_sync -d /my_documents -r colleague@B:/home/my_documents -w pass -K bidi
/              SYNCHRONIZED
/Document1     SYNCHRONIZED
/Document2     SYNCHRONIZED
/Document3     SYNCHRONIZED
```

2. Delete the conflicted file from one endpoint

A faster method, but it does not preserve changes on both sides and requires access to both endpoints.

If you have access to the file on both endpoints, compare the files and determine if the changes to the conflicted file on one endpoint do not need to be preserved (such as if they duplicate changes on the other endpoint or they add obsolete or incorrect information). If changes on both endpoints need to be preserved, use one of the other methods.

Delete the file that has changes you do not want to preserve and run the Sync session again. The version with the changes you want to keep is synchronized between the two endpoints. For example, if the changes to `Document2` on computer B do not need to be preserved, delete `Document2` on computer B and then run the session again. All files are synchronized.

3. Rename the conflicted file on one side

The fastest method, changes on both sides are preserved but in separate files, allowing you to resolve the original conflict after synchronization. Requires access to only one endpoint.

If you only have access to one endpoint, want to preserve changes on both sides, but do not want to resolve the conflict immediately, you can rename the conflicted file on one endpoint. When you run the **async** session, both endpoints have the two versions of the conflicted file. You can then compare the differences between them and make edits to the original file later.

For example, rename `Document2` on computer A to `Document2_computerA`. When you run the **async** session, computer A and computer B both have the following files:

```
async -N my_bidi_sync -d /my_documents -r colleague@B:/home/my_documents -w pass -K bidi
/              SYNCHRONIZED
/Document1     SYNCHRONIZED
/Document2     SYNCHRONIZED
/Document2_computerA    SYNCHRONIZED
/Document3     SYNCHRONIZED
```

Appendix

Hardlinks

On Unix-based systems, it's possible to encounter multiple files with the same inode. The most common case of this is a hardlink. Sync is agnostic as to whether two files with multiple inodes are hardlinks or if they are actually different. It assumes that directories have unique inodes.

Handling Hardlinks

- One or more hardlinks at the source become regular files at the destination.
- In continuous mode, if a file with multiple links changes, only that file is replicated at the destination (even though all links at the source changed).
- In scan mode (one-time and continuous startup), all files for that link are replicated at the destination.

Handling Moves in Scan Mode

- If a new file has only one link, Sync checks whether it is a move.
- If a new file has two or more links, Sync does not check whether it is a move (regardless of whether the other links are inside or outside the Syncdirectory).
- For directories, Sync checks whether or not it is a move.

Handling Moves in Continuous Mode

For all files and directories, notifications tell Sync the old and new paths; thus, a move is recognized in all cases.

Creating SSH Keys

About this task

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.

If you are using this machine as a client to connect to other Aspera servers with public key authentication, you need to generate a key-pair for the selected user account, as follows:

Procedure

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. Generate an SSH key-pair.

   In the `.ssh` folder, use the **ssh-keygen** command to create a key pair.

   ```
   # ssh-keygen -m key_format -t key_type
   ```

   - For *key_format*, specify a format that is supported by the SSH server.
   - For *key_type*, specify either RSA (`rsa`) or ECDSA (`ecdsa`).

At the prompt that appears 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.

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 pkhsts8 -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. The instructions for installing the public key on the server can be found in the "Setting Up a User's Public Key on the Server" on page 24; however, the server may be installed on an operating system that is different from the one where your client has been installed.

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`.

rsync vs. async Uni-directional Example

The **async** and **rsync** command-line options are similar for basic uni-direction synchronization.

Below are examples of **rsync** commands and their **async** equivalents for uni-directional synchronization.

Example 1

Options:

- Recursively synchronize the contents of a directory, `/media/` to the remote system directory `/backups/media`
- Preserve access and modification time stamps on files
- Preserve the owner and group ID
- No encryption
- Transfer policy = fair
- Target rate = 100,000 Kbps (100 Mbps)
- One-time transfer (not continuous)

rsync command:

```
# rsync --stats -v -r -u -l -t -o -g -p /media/ editor@docserver:/backups/media
```

async equivalent:

```
# async -N Oneway -u -t -j -d /media/ -r editor@docserver:/backups/media -l 100M -w d0c5 -K
push -c none
```

Example 2

Options:

- Recursively synchronize the contents of the directory `/media/wmv/`
- Exclude "." files within the directory
- Exclude all other directories
- Preserve the owner and group ID
- Preserve access and modification time stamps on files
- No encryption
- Transfer policy = fair
- Target rate = 100,000 Kbps (100 Mbps)
- One-time transfer (not continuous)

rsync command:

```
# rsync --stats -v -r -u -l -t -o -g -p /media/ --include="/media" --include="/media/wmv" --
exclude="/media/.*" editor@docserver:/backups/media
```

async equivalent:

```
# async -N Oneway -u -t -j -d /media/ --include="/media" --include="/media/wmv" --exclude="/
media/.*" -r editor@docserver:/backups/media -w d0c5 -K push -c none
```

Options Comparison Table

rsync Option	**async Option**	**Description**
--stats	Enabled by default	Display file transfer status
-v, --verbose	Enabled by default	Increase verbosity
-q, --quiet	-q, --quiet	Disable progress display
-r, --recursive	Enabled by default	Recurse into directories
-u, --update	If a file exists at the destination with the same name, then the default behavior is to do nothing if the files are the same (size and checksum), and overwrite if the file is different.	Skip files that are newer on the receiver
-l, --links	Linux and macOS: `-n copy`, `--symbolic-links=copy` Symbolic links are skipped in Windows.	Copy symbolic links as symbolic links (Linux and macOS only)
-t, --times	-t, --preserve-time (must have HSTS or High-Speed Transfer Endpoint 3.1+)	Preserve modification times
-o, --owner	-u, --preserve-uid	Preserve owner
-g, --group	-j, --preserve-gid	Preserve group
-p, --perms	With regard to directory attributes, if the source mode doesn't have sufficient owner permissions, then the destination will add: owner rwx.	Preserve permissions
--version	-A, --version	Print version number

rsync Option	**async Option**	**Description**
-h, --help	-h, --help	Show help
--include-from=*file*	-I, --include-from=*file*	Include filter (text file with paths for inclusion). See “Using Filters to Include and Exclude Files” on page 145.
--exclude-from=*file*	-E, --exclude-from=*file*	Exclude filter (text file with paths for exclusions). See “Using Filters to Include and Exclude Files” on page 145.
--include=*pattern*	--include=*pattern*	Include paths that match *pattern*. See “Using Filters to Include and Exclude Files” on page 145.
--exclude=*pattern*	--exclude=*pattern*	Skip paths that match *pattern*. See “Using Filters to Include and Exclude Files” on page 145.
	-c *none*	**rsync**, as a protocol, does not encrypt on its own; however, **rsync** can enable/disable the SSH encryption protocol (using option -e *ssh*).

Configuring for Other Aspera Products

HSTS can be configured as the transfer server for IBM Aspera Faspex, IBM Aspera Shares, and IBM Aspera Application for Microsoft SharePoint. It can also be configured and added as a node to Shares, IBM Aspera on Cloud (AoC), and IBM Aspera Console.

For instructions on how to configure HSTS for Aspera web applications, see their Admin guides:

- **AoC:** https://ibm.ibmaspera.com/helpcenter/admin/nodes/configuring-an-aspera-transfer-server-as-a-node-for-aspera-on-cloud
- **Console:** IBM Aspera Console Admin Guide
- **Faspex:** IBM Aspera Faspex Admin Guide
- **Shares:** IBM Aspera Shares Admin Guide
- **Aspera for SharePoint:** IBM Aspera Application for Microsoft SharePoint Admin Guide

Set up HSTS for Node API

HSTS must be configured in order to use the Aspera Node API. You can use the **asnodeadmin** tool to set up the server and manage the Node API. The Node API uses a Redis database, which can be backed up and restored in different ways, depending on what information you need to preserve.

Overview: Aspera Node API

The Aspera Node API is a feature of HSTS that provides a REST API for full programmatic control of the Aspera transfer server environment. The asperanoded daemon (which is run by the asperadaemon user), provides node-specific services such as browsing, searching, creating and deleting files and directories, and setting up transfers over HTTP or HTTPS.

The Node API allows you to connect nodes to Aspera web applications, such as IBM Aspera Faspex, IBM Aspera Shares, and IBM Aspera on Cloud, as well as integrate Aspera applications with your web application. It is supported by all Aspera server products and across multi-cloud and hybrid storage systems.

Note: If you are going to expose asperanoded to the internet, Aspera strongly recommends that you increase security by using a proxy. See “Securing the Node Service Behind a Reverse Proxy” on page 302.

The Node API includes the following features and functionality:

- An HTTPS (by default port 9092) and HTTP (by default port 9091) interface.
- An API that uses JSON data format.
- The API is authenticated and the node daemon uses its own application-level users (*node users*).
- A node admin utility, **asnodeadmin**, for adding and managing Node API users and passwords. For more information, see "Node Admin Tool" on page 296.
- It logs to `syslog`, akin to `asperacentral`.

Requirements to use the Node API:

- The line `127.0.0.1 localhost` must appear in the `hosts` file:

  `/etc/hosts`
- For UNIX-based nodes, SELinux must be set to `permissive` or `disabled`, not `enforced`. Check the status of SELinux with the following command:

  ```
  # sestatus
  ```

  If SELinux is set to `enforced`, see "Disabling SELinux" on page 344. If SELinux is set to `disabled` or `enforced`, or if **sestatus** reports that SELinux is not installed on your system, you do not need to take further action.
- To run node-to-node transfers, the remote node must have version 3.7.4 or later. Earlier versions use an SSH key type that is no longer accepted by servers as of version 3.7.4.

Node API Setup

The Aspera Node API comes with your installation of HSTS. To use it, you must configure your product and create Node API credentials.

Procedure

1. Ensure that the hosts file contains an entry for `127.0.0.1 localhost`. The hosts file can be found in `/etc/hosts`.
2. Select or create a system user to associate with the Node API credentials.

   Aspera uses a specially configured system user for SSH authentication when starting transfers.

   Note: If this user will be associated with Node API credentials that will be used to create access keys or bearer tokens, either do not set a password for the user or create a very large password.

   Create a user account—for example, `aspera_user_1`—by running the following command:

   ```
   # useradd aspera_user_1
   ```
3. Restrict the system user's access to the server's file system.

   If the Node API user will use access key or bearer token authentication to authenticate to the Node API, configure a restriction for the system user. If the Node API user will use Node API credentials to authenticate to the Node API, configure a docroot for the system user.

   - **To configure a restriction:**

     Run the following command:

     ```
     # asconfigurator -x "set_user_data;user_name,username;file_restriction,|restriction"
     ```

     Where *username* is the system user's username, | is a delimiter, and *restriction* is specific to the storage type and path:

Storage Type	Format Example
local storage	For Unix-like OS: – specific folder: `file:////folder/*` – drive root: `file:////*` For Windows OS: – specific folder: `file:///c%3A/folder/*` – drive root: `file:///c*`
Amazon S3 and IBM Cloud Object Storage - S3	`s3://*`
Azure	`azu://*`
Azure Files	`azure-files://*`
Azure Data Lake Storage	`adl://*`
Alibaba Cloud	`oss://*`
Google Cloud	`gs://*`
HDFS	`hdfs://*`

- **To configure a docroot:**

  Run the following command:

  ```
  # asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
  ```

  Where *username* is the system user's username and *docroot* is the absolute path to which the system user has access.

4. Restrict user permissions with **aspshell**.

   By default, all system users can establish a FASP connection and are only restricted by file permissions. Restrict the user's file operations by assigning them to use **aspshell**, which permits only the following operations:

   - Running Aspera uploads and downloads to or from this computer.
   - Establishing connections in the application.
   - Browsing, listing, creating, renaming, or deleting contents.

   These instructions explain one way to change a user account or active directory user account so that it uses the **aspshell**; there may be other ways to do so on your system.

   Run the following command to change the user login shell to **aspshell**:

   ```
   # sudo usermod -s /bin/aspshell username
   ```

   Confirm that the user's shell updated by running the following command and looking for `/bin/aspshell` at the end of the output:

   ```
   # grep username /etc/passwd
   username:x:501:501:...:/home/username:/bin/aspshell
   ```

   Note: If you use OpenSSH, sssd, and Active Directory for authentication: To make `aspshell` the default shell for all domain users, first set up a local account for server administration because this change affects all domain users. Then open `/etc/sssd/sssd.conf` and change `default_shell` from `/bin/bash` to `/bin/aspshell`.

5. Set the IBM Aspera Connect public SSH key as an authorized key for the transfer user and ensure that they own the file.

   a) Create the `.ssh` directory in the user's home folder.

```
# mkdir /home/aspera_user_1/.ssh/
```

b) Copy the Connect public SSH key into `.ssh` and rename it `authorized_keys` (or append the public key to `authorized_keys` if the file already exists).

```
# cp /opt/aspera/var/aspera_tokenauth_id_rsa.pub /home/aspera_user_1/.ssh/authorized_keys
```

c) Ensure that `.ssh` and `.ssh/authorized_keys` are owned by the user.

```
# chown -R aspera_user_1:aspera_user_1 /home/aspera_user_1/.ssh
# chmod 600 /home/aspera_user_1/.ssh/authorized_keys # chmod 700 /home/aspera_user_1

# chmod 700 /home/aspera_user_1/.ssh
```

6. Associate the Aspera transfer user with a Node API username and password.

   For example, to assign Node API credentials to user `aspera_user_1`, run the following command:

```
# /opt/aspera/bin/asnodeadmin -a -u node_api_username -p node_api_passwd -x aspera_user_1
```

7. (Optional) Change HTTPS port and/or SSL certificate.

   The Aspera Node API provides an HTTPS interface for encrypted communication between node machines (on port 9092, by default). To modify the HTTPS port, see "Configuring the IBM Aspera NodeD Service" on page 297. For information on maintaining and generating a new SSL certificate, see "Setting up SSL for your Nodes" on page 304.

8. Configure other Node API settings.

   - If you want to query transfers by using GET /ops/transfers or to retrieve usage data by using GET /usage, enable activity logging on the node by running the following command:

```
# asconfigurator -x "set_server_data;activity_logging,true"
```

   - If you want to query events by using GET /events, enable activity event logging on the node by running the following command:

```
# asconfigurator -x "set_server_data;activity_event_logging,true"
```

     As of version 3.8.0, `activity_event_logging` can be configured in individual access keys and overrides the setting on the node. If `activity_event_logging` is enabled for the access key, any Node API events associated with that access key are logged even if the node setting is false. If it is disabled for the access key, events are not logged for the access key even if `activity_event_logging` is enabled on the node.

   - For a description of other settings, see "Configuring the IBM Aspera NodeD Service" on page 297.

9. Restart asperanoded to activate your changes.

   Run the following commands to restart asperanoded:

```
# systemctl restart asperanoded
```

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

```
# service asperanoded restart
```

Node Admin Tool

Use the **`asnodeadmin`** tool to manage (add, modify, delete, and list) Node API users. Root privileges are required.

Syntax:

```
# /opt/aspera/bin/asnodeadmin [options]
```

For a complete list of options, use:

```
asnodeadmin -h
```

Usage Examples

1. Add Node API username **usr1** with the Node API password **pass1** (you are prompted to enter if the `-p` option is not given) and associate them with the transfer user **aspera**:

```
# /opt/aspera/bin/asnodeadmin -au usr1 -x aspera [-p pass1]
```

2. Add Node API username **usr2** with Node API password **pass2** and associate them with transfer user **root**:

```
# /opt/aspera/bin/asnodeadmin -au usr2 -p pass2 -x root
```

3. Modify Node API username **usr1** by assigning a different password, **pass1.1**:

```
# /opt/aspera/bin/asnodeadmin -mu usr1 -p pass1.1
```

4. List Node API usernames in the current user database:

```
# /opt/aspera/bin/asnodeadmin -l
```

5. Delete Node API username **usr1**:

```
# /opt/aspera/bin/asnodeadmin -du usr1
```

6. Create a bearer token: See "Bearer Tokens" on page 324.

Configuring the IBM Aspera NodeD Service

The IBM Aspera NodeD Service handles HTTP/HTTPS requests to HSTS. You can configure server settings including the hostname, HTTP/HTTPS ports, the address and port of the Redis database, and SSL certificates.

Configuration Methods

The server can be configured for the Node API by using the **`asconfigurator`** command-line tool or by editing the `<server>` section of `aspera.conf`:

- **Asconfigurator:** Use the following syntax, substituting *option* with the option from the following table and *value* with the desired value:

  ```
  # /opt/aspera/bin/asconfigurator -x "set_server_data;option,value"
  ```

  To view the current settings, run the following command:

  ```
  # /opt/aspera/bin/ asuserdata -a
  ```

- **Aspera.conf:** Open it in a text editor with administrative privileges from the following location:

  ```
  /opt/aspera/etc/aspera.conf
  ```

  See the sample `aspera.conf` following the table.

  After manually editing `aspera.conf`, validate your XML by running the following command:

  ```
  # /opt/aspera/bin/asuserdata -v
  ```

Node API Configuration Options

Important configuration considerations:

- Certain services must be restarted for changes in the settings to take effect, as described in the **To Activate Changes** column. The commands to restart these services are given following the table.
- In addition to the Aspera server configuration, if you plan to transfer many small files with the Node API, you might need to increase the number of file descriptors available on your system. If too few descriptors are available, the Redis database and the transfer fail. For instructions, see "Node API Transfers of Many Small Files Fails" on page 343.

asconfigurator option **aspera.conf setting**	**Description and Values**	**To Activate Changes...**
`server_name` `<server_name>`	Hostname or IP address. Default: *hostname*	Restart asperanoded
`http_port` `<http_port>`	HTTP service port. Value is an integer 1 - 65535, default 9091. This setting is overridden by `<listen>`.	Restart asperanoded
`https_port` `<https_port>`	HTTPS service port. Value is an integer 1 - 65535, default 9092. This setting is overridden by `<listen>`.	Restart asperanoded
`enable_http` `<enable_http>`	Enable HTTP for the Node API services by setting to `true`. Default: `false`. This setting is overridden by `<listen>`.	Restart asperanoded
`enable_https` `<enable_https>`	Enable HTTPS for the Node API services by setting to `true` (default). This setting is overridden by `<listen>`.	Restart asperanoded
`workers` `<workers>`	Number of worker threads. Default: 20.	Restart asperanoded
`transfers_multi_session_default` `<transfers_multi_session_default>`	Number of **ascp** workers per transfer. Default: 1.	Restart asperanoded
`transfers_retry_duration` `<transfers_retry_duration>`	If a transfer fails, node will try to restart it for the specified time, default 20m. If a transfer restarts and makes some progress, then the retry timer is reset and the next time if fails, it will again try to restart it for 'retry_duration'. The backoff interval for retrying within this duration is internal to the application, and the number of retries may vary depending on the transfer queue.	Restart asperanoded
`transfers_retry_all_failures` `<transfers_retry_all_failures>`	Setting to `true` will retry all transfers, including transfers otherwise considered unretriable. Default: `false`.	Restart asperanoded
`listen` `<listen>`	To bind **asperanoded** on a specific address (or addresses), specify a comma-delimited list of listening ports. Ports have the format `[ip_address:]port[s]`. To specify a secure port, add 's' to the end of the port number, for example `127.0.0.1:9092s`.	Restart asperanoded

asconfigurator option aspera.conf setting	Description and Values	To Activate Changes...
	The IP address is optional; however, if no IP address is specified then the port binds to all network interfaces on the server, rather than to the single address. Setting this option overrides <http_port>, <https_port>, <enable_http>, and <enable_https>.	
cert_file <cert_file>	Full pathname of the SSL certificate, which must be in .pem format. Default: /opt/aspera/etc/aspera_server_cert.pem	Restart asperanoded
max_response_entries <max_response_entries>	Maximum number of entries to return in a response. Default: 1000.	Reload node configuration.
max_response_time <max_response_time>	Maximum amount of time to wait for a long-running operation. Default: 10.	Reload node configuration.
db_dir <db_dir>	Path to the directory where the database file is saved. Before changing this value, you should back up your database. See “Backing up and Restoring the Node User Database Records” on page 302. Default: /opt/aspera/var	Restart asperanoded and the Redis database
db_port <db_port>	Database service port. Value is an integer 1 - 65535, default: 31415. Before changing this value, you should back up your database. See “Backing up and Restoring the Node User Database Records” on page 302.	Restart asperanoded and the Redis database
ssl_ciphers <ssl_ciphers>	The SSL encryption ciphers that the server will allow, each separated by a colon (:). Default: all of the following: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA DHE-RSA-AES256-SHA DHE-DSS-AES256-SHA AES256-SHA AES128-SHA256 DHE-RSA-AES128-SHA DHE-DSS-AES128-SHA RC2-CBC-MD5	Restart asperanoded

asconfigurator option aspera.conf setting	Description and Values	To Activate Changes...
	This option may also be set in the <client> section, in which case, when this machine functions as a client, the specified ciphers are requests to the server. If any of the ciphers in the server's allow list coincide with those in the client's request list, communication is allowed; otherwise it is denied. If you override this setting, the override is always used. However, if you do not override it, the default setting depends on the settings for <ssl_protocol>. If <ssl_protocol> is set to sslv23, then a large, relatively weak selection of suites is allowed. If the protocol is anything else, then a smaller, stronger selection of suites is allowed. Many older web browsers cannot handle the stronger set of suites, in which case you may encounter compatibility issues.	
ssl_protocol <ssl_protocol>	The SSL protocol versions that the server will allow. This option may also be set in the <client> section, in which case, when this machine is a client, the specified protocols function as requests to the server. If any of the protocols in the server's allow list coincide with those in the client's request list, communication is allowed; otherwise it is denied. Supported values: tlsv1, tlsv1.1, and tlsv1.2. Default: tlsv1.	Restart asperanoded
activity_logging <activity_logging>	If true, enable querying transfers by using GET /ops/transfers or to retrieve usage data by using GET /usage. Default is false.	Restart asperanoded
activity_event_logging <activity_event_logging>	If true, allow the Node API to query transfers that are associated with this access key through the /events endpoint. The server configuration can be overridden by the access key configuration. This option must be enabled for event reporting to IBM Aspera on Cloud. Default is false.	Restart asperanoded
files_recursive_counts_enabled <files_recursive_counts_enabled>	If true, enable recursive counts. This option must be enabled for event reporting to IBM Aspera on Cloud. The server configuration can be overridden by the access key configuration. Default is false.	Restart asperanoded
aej_logging <aej_logging>	If true, enable reporting to the IBM Aspera on Cloud Activity app. The server	Restart asperanoded

asconfigurator option aspera.conf setting	Description and Values	To Activate Changes...
	configuration can be overridden by the access key configuration. Default is false.	

Example Node API Configuration in `aspera.conf`

```
<server>
   <server_name>your_hostname</server_name>
   <http_port>9091</http_port>
   <https_port>9092</https_port>
   <enable_http>false</enable_http>
   <enable_https>true</enable_https>
   <workers>20</workers>
   <transfers_multi_session_default>1</transfers_multi_session_default>
   <transfers_retry_all_failures>false</transfers_retry_all_failures>
   <transfers_retry_duration>20m</transfers_retry_duration>
   <listen> </listen>
   <cert_file>/opt/aspera/etc/aspera_server_cert.pem</cert_file>
   <max_response_entries>1000</max_response_entries>
   <max_response_time_sec>10</max_response_time_sec>
   <db_dir>/opt/aspera/var</db_dir>
   <db_port>31415</db_port>
   <ssl_ciphers>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA:…:RC2-CBC-MD5</ssl_ciphers>
   <ssl_protocol>tlsv1</ssl_protocol>
   <activity_logging>true</activity_logging>
   <activity_event_logging>true</activity_event_logging>
   <files_recursive_counts_enabled>true</files_recursive_counts_enabled>
   <aej_logging>true</aej_logging>
</server>
```

Restarting and Reloading Services

Note: Running the commands below requires root privileges.

Restart asperanoded:

Run the following commands to restart asperanoded:

```
# systemctl restart asperanoded
```

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

```
# service asperanoded restart
```

Reload the Node Configuration:

```
# sudo /opt/aspera/bin/asnodeadmin --reload
```

Restart asperanoded and the Redis database:

1. Stop asperanoded:

   ```
   # systemctl stop asperanoded
   ```

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

   ```
   # service asperanoded stop
   ```

2. Shutdown the database:

   ```
   # /opt/aspera/bin/asnodeadmin --db-shutdown
   ```

3. Start asperanoded:

```
# systemctl start asperanoded
```

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

```
# service asperanoded start
```

Note: The database service is started automatically when you restart the node service.

Securing the Node Service Behind a Reverse Proxy

If you are going to expose asperanoded to the internet, Aspera strongly recommends that you use a reverse proxy . For example, if you want to use an HSTS instance with IBM Aspera on Cloud (AoC)—that is, as a user-managed *tethered node*— you should use a reverse proxy with it.

Backing up and Restoring the Node User Database Records

These instructions describe how to back up and restore your Node API user data up to the time of the backup operation.

Procedure

1. Back up the Node API user data from the Redis database:

   ```
   # sudo /opt/aspera/bin/asnodeadmin -b /filepath/database.backup
   ```

   Important: When backing up the Redis database, all user data up to that point in time will be saved to the backup file. *Restoring the database (see Step 2, below) does not delete users added after this snapshot was taken.* Thus, if you added any users after backing up the database, they still exist in the system and are not affected by the restore operation.
2. Restore the Node API user data to the Redis database:

   ```
   # sudo /opt/aspera/bin/asnodeadmin -r /filepath/database.backup
   ```

   Note: If you do not want to keep users that have been added since the last backup operation, delete them after performing the restore with the following command:

   ```
   # sudo /opt/aspera/bin/asnodeadmin -du username
   ```
3. Restart asperanoded:

   Run the following commands to restart asperanoded:

   ```
   # systemctl restart asperanoded
   ```

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

   ```
   # service asperanoded restart
   ```

Backing up and Restoring Access Keys (Tenant Data)

Access keys can be backed up and restored by using the **asnodeadmin** tool. Only master access keys can be directly backed up, not sub-access keys, but backing up a master access key backs up all associated sub-access keys, too.

Access keys are not backed up when you back up the Node API user database (“Backing up and Restoring the Node User Database Records” on page 302), but they are if you back up the entire Redis database (“Backing up and Restoring a Node Database” on page 303).

Back up Access Keys

Run the following command for each access key:

```
# /opt/aspera/bin/asnodeadmin --access-key access_key_id --access-key-backup filename
```

Where *filename* is the AOF file to which the access key data is saved.

Restore Access Keys

Run the following command:

```
# /opt/aspera/bin/asnodeadmin [-u username] --access-key-restore filename
```

Use the `-u username` option to change the Node API user (and system user) associated with the restored access key.

Backing up and Restoring a Node Database

These instructions describe how to back up and restore the entire Redis database of a node, which includes Node API users, their access keys, and transfer history. If your transfer server is an IBM Aspera on Cloud (AoC) node, migrate AoC data from one node to another by backing up the Redis database on the original node and restoring the database on a new node.

About this task

If you only need to back up and restore Node API usernames and passwords (the Node API user database), use **asnodeadmin** commands; see "Backing up and Restoring the Node User Database Records" on page 302. If you also want to back up and restore access keys, see "Backing up and Restoring Access Keys (Tenant Data)" on page 302.

These instructions assume that the node is using the default port for the Redis database, port 31415. If your deployment uses a different port for Redis, substitute it in the commands accordingly.

Procedure

1. Verify that the original node and new node are running the same version of Aspera software.
   Run **ascp -A** on a command line to view the Aspera product and version.
2. On the original node, back up the database.
   Stop asperanoded and create the backup file by running the following commands:

   ```
   # systemctl stop asperanoded
   # /opt/aspera/bin/asredis -p 31415 BGREWRITEAOF
   ```

   The backup is stored as `appendonly.aof` in the following location:

   ```
   /opt/aspera/var/appendonly.aof
   ```
3. If migrating the database, move the `appendonly.aof` to the same location on the new node.
4. On the new node, stop asperanoded:

   ```
   # systemctl stop asperanoded
   ```
5. Flush existing data from the Redis database on the new node.

   ```
   #/opt/aspera/bin/asredis -p 31415 FLUSHALL
   ```
6. Load the backup database file into the new node database.

   ```
   # cat appendonly.aof | /opt/aspera/bin/asredis --pipe -p 31415
   ```

7. On both nodes, restart asperanoded.

```
# systemctl start asperanoded
```

8. In AoC, confirm that the hostname matches the DNS entry for the new node.

   To view the node URL, go to **Admin View > Nodes & Storage**.

9. Confirm the database restoration succeeded.

   Run the following command to the original and new nodes. If the database restoration succeeded, the output from each is identical.

```
# curl -ki -u {node_username:node_password} http[s]://{hostname}:{http_port}access_keys
```

   Note: Curl is included in many Unix-based operating systems. To check if it is installed, enter **curl** on the command line. If it is not installed, download it from the Curl website: https://curl.haxx.se/download.html.

Setting up SSL for your Nodes

The Aspera Node API provides an HTTPS interface for encrypted communication between nodes (on port 9092, by default). For example, if you are running the IBM Aspera Faspex web UI or the IBM Aspera Shares web UI on one computer, you can encrypt the connection (using SSL) with your transfer server or file-storage node on another computer. HSTS nodes are preconfigured to use Aspera's default, self-signed certificate (`aspera_server_cert.pem`). You might need to create a new certificate or install a valid, signed certificate, such as when you are configuring HSTS as a IBM Aspera on Cloud node.

About this task

The self-signed Aspera certificate is located in the following directory:

```
/opt/aspera/etc/
```

About PEM Files: The PEM certificate format is commonly issued by Certificate Authorities. PEM certificates have extensions that include .pem, .crt, .cer, and .key, and are Base-64 encoded ASCII files containing "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" statements. Server certificates, intermediate certificates, and private keys can all be put into the PEM format.

To generate a new certificate:

Procedure

1. Generate a Private Key and Certificate Signing Request (CSR) using OpenSSL.

   In a Terminal window, run the following command (where `my_key_name.key` is the name of the unique key that you are creating and `my_csr_name.csr` is the name of your CSR):

```
# openssl req -new -nodes -keyout my_key_name.key -out my_csr_name.csr
```

2. At the prompt, enter your X.509 certificate attributes.

   Important: The Common Name field must be filled in with the fully qualified domain name of the server to be protected by SSL. If you are generating a certificate for an organization outside the U.S., go to https://www.iso.org/obp/ui/, select **Country codes**, and click  to view a list of two-letter ISO country codes.

```
Generating a 1024 bit RSA private key
...................++++++
.............++++++
writing new private key to 'my_key_name.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:Your_2_letter_ISO_country_code
State or Province Name (full name) [Some-State]:Your_State_Province_or_County
Locality Name (eg, city) []:Your_City
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your_Company
Organizational Unit Name (eg, section) []:Your_Department
Common Name (i.e., your server's hostname) []:secure.yourwebsite.com
Email Address []:johndoe@yourwebsite.com
```

You are also prompted to input "extra" attributes, including an optional *challenge password*.

Note: Manually entering a challenge password when starting the server can be problematic in some situations, for example, when starting the server from the system boot scripts. Skip entering a challenge password by pressing **Enter**.

```
...
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

After finalizing the attributes, the private key and CSR are saved to your root directory.

Important: If you make a mistake when running the OpenSSL command, you may discard the generated files and run the command again. After successfully generating your key and CSR, be sure to guard your private key, as it cannot be re-generated.

3. If required, send the CSR to your Certifying Authority (CA).

   Once completed, you have a valid, signed certificate.

   Note: Some certificate authorities provide a CSR generation tool on their website. For additional information, check with your CA.

4. If required, generate a self-signed certificate.

   You may need to generate a self-signed certificate for the following reasons:

   - You don't plan on having your certificate signed by a CA.
   - You plan to test your new SSL implementation while the CA is signing your certificate.

   To generate a self-signed certificate through OpenSSL, run the following command:

   ```
   # openssl x509 -req -days 365 -in my_csr_name.csr -signkey my_key_name.key -out
   my_cert_name.crt
   ```

   This creates a certificate that is valid for 365 days.

5. Create the `.pem` file.

   Note: Before overwriting the existing `.pem` file, be sure to back up this file as `aspera_server_cert.old`), in the following directory:

   ```
   /opt/aspera/etc/
   ```

   Copy and paste the entire body of the key and cert files into a single text file and save the file as `aspera_server_cert.pem`. The order of the text in the new `.pem` file depends on if you have individual certificate files or a bundle of certificates.

   Individual certificate files:

   a. The private key.

   b. The primary server's certificate.

   c. The intermediate certificates, if any (if more than one, begin with the least authoritative and proceed in ascending order).

   d. The root certificate.

   Bundle of certificates:

a. The private key.

b. The primary server's certificate.

c. The entire bundle (as one file).

For a certificate bundle, create a new file named `aspera_server_cert.chain` in the same directory as the `.pem` files. Copy and paste the root certificate into this file, followed by the bundle.

6. Enable SSL options in `aspera.conf`.

   For information about enabling specific SSL protocols with `<ssl_protocol>` and enabling specific encryption ciphers with `<ssl_ciphers>`, see "Configuring the IBM Aspera NodeD Service" on page 297.

7. Restart asperanoded by running the following command:

   Run the following commands to restart asperanoded:

   ```
   # systemctl restart asperanoded
   ```

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

   ```
   # service asperanoded restart
   ```

Installing SSL Certificates

Aspera strongly recommends installing valid, signed SSL certificates on your HSTS. The SSL certificates are asperanoded and asperahttpd.

About this task

Requirements:

- A signed root certificate or certificate bundle (root certificate with chained or intermediary certificates) from an authorized Certificate Authority. For instructions on generating an SSL certificate, see "Setting up SSL for your Nodes" on page 304.
- The certificate is in `.pem` format. Other formats are not supported.

Procedure Overview:

The procedure modifies or creates three files:

aspera_server_key.pem

- Created automatically during transfer server installation.
- Found in the default Aspera installation directory: `/opt/aspera/etc`
- Contains the default private key.
- In this procedure, you replace the default private key with the new private key generated with the certificate signing request (CSR).

aspera_server_cert.pem

- Created automatically during transfer server installation.
- Found in the default Aspera installation directory: `/opt/aspera/etc`
- Contains the default self-signed certificate.
- In this procedure, you replace the default self-signed certificate with the content described in step 3.

aspera_server_cert.chain

- You create this file, as described below.
- You place the file in the same directory as `aspera_server_key.pem` and `aspera_server_cert.pem`.

- You place the certificate bundle (chained or intermediary certificates) from the CA in this file.

Changing Filenames and Locations:

If desired, the default filenames and locations of the certificate files and chain files can be changed by configuring settings in the transfer server's `aspera.conf` file, using **`asconfigurator`** commands:

```
# asconfigurator -x "set_http_server_data;cert_file,path/certfile.pem"
# asconfigurator -x "set_http_server_data;key_file,path/keyfile.pem"
# asconfigurator -x "set_server_data;cert_file,path/certfile.pem"
```

Note: The chain file for `asperanoded` must match the location and name of the `asperanoded` certificate file, but with the `.chain` extension.

The commands add the following text to `aspera.conf`:

```
<http_server>
    ...
    <key_file>path/keyfile.pem</key_file>     <!-- key file for asperahttpd -->
    <cert_file>path/certfile.pem</cert_file>  <!-- cert file for asperahttpd -->
    ...
</http_server>

<server>
    ...
    <cert_file>path/certfile.pem</cert_file>  <!-- cert file for asperanoded -->
    ...
</server>
```

Installing the SSL Certificates:

Procedure

1. Back up the default private key and self-signed certificate, using the following commands:

   ```
   # cd /opt/aspera/etc
   # cp aspera_server_key.pem aspera_server_key.pem.bak
   # cp aspera_server_cert.pem aspera_server_cert.pem.bak
   ```

2. Open `aspera_server_key.pem` and replace the existing content with the new private key generated with the certificate signing request (CSR). Save and close the file.
3. In `aspera_server_cert.pem`, replace the existing content with the following, in the order shown:
   a. the new private key
   b. the server certificate
   c. any chained or intermediary certificates from the CA in order of ascending authority, for example:

      intermediary certificate 1
      intermediary certificate 2
      intermediary certificate 3

   d. the root certificate from the CA

   Save and close the file.
4. Create a new file named `aspera_server_cert.chain`. This file must reside in the same directory as the `.pem` files.

   If you *have* a certificates bundle from the CA, the contents of `aspera_server_cert.chain` must consist of the following, in the order shown:

   a. the server certificate
   b. the certificates bundle, which includes the root certificate

   If you do not have a certificates bundle from the CA, the contents of `aspera_server_cert.chain` must consist of the following, in the order shown:

   a. the server certificate

b. any chained or intermediary certificates from the CA in order of ascending authority, for example:

   intermediary certificate 1
   intermediary certificate 2
   intermediary certificate 3

c. the root certificate from the CA

5. Restart asperacentral, asperanoded, and asperahttpd:

```
# service asperacentral restart
# service asperahttpd restart
# service asperanoded restart
```

6. Verify the certificates by using OpenSSL.

a) Test that you can connect to asperanoded by running the following command:

```
# /opt/aspera/bin/openssl s_client -connect myserver:9092
```

This example assumes that you are using the default node port (HTTPS 9092). Replace *myserver* with the IP address or hostname of your server.

The command returns 0 for success or 1 for failure.

Output examples:

Success: The following sample output shows that verification was successful because `verify return` is 0.

```
depth=2 C = US, O = "VeriSign, Inc.", OU = VeriSign Trust Network, OU = "(c) 2006
VeriSign, Inc. -
For authorized use only", CN = VeriSign Class 3Public Primary Certification Authority - G5
verify error:num=20:unable to get local issuer certificate
verify return:0
```

Failure: The following sample output shows that verification failed because `verify return` is 1.

```
depth=0 C = US, ST = California, L = Emeryville, O = IBM, OU = Aspera Inc IT Department,
CN = *.asperafiles.com
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 C = US, ST = California, L = Emeryville, O = IBM, OU = Aspera Inc IT Department,
CN = *.asperafiles.com
verify error:num=27:certificate not trusted
verify return:1
depth=0 C = US, ST = California, L = Emeryville, O = IBM, OU = Aspera Inc IT Department,
CN = *.asperafiles.com
verify error:num=21:unable to verify the first certificate
verify return:1
```

Note: You must see as many elements in the output as there are certificates in the chain. In the following examples there is one root certificate and two chained certificates, so the output must show three elements to prove the installation was successful.

Success: The following example shows a successful verification for one root certificate and two intermediary certificates in the chain:

```
Certificate chain
 0 s:/C=US/ST=California/L=Emeryville/O=IBM/OU=Aspera Inc IT Department/
CN=*.asperafiles.com
 i:/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure
Server CA - G4
 1 s:/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure
Server CA - G4
 i:/C=US/O=VeriSign, Inc./OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - For
authorized use only/CN=VeriSign Class 3 Public Primary Certification Authority - G5
 2 s:/C=US/O=VeriSign, Inc./OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - For
authorized use only/CN=VeriSign Class 3 Public Primary Certification Authority - G5
 i:/C=US/O=VeriSign, Inc./OU=Class 3 Public Primary Certification Authority
```

Failure: The following example shows an unsuccessful verification, since only the root certificate is displayed.

```
Certificate chain
 0 s:/C=US/ST=California/L=Emeryville/O=IBM/OU=Aspera Inc IT Department/
CN=*.asperafiles.com
 i:/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure
Server CA - G4
```

b) If verification fails, inspect your certificate content by running the following command:

```
# /opt/aspera/bin/openssl x509 -in certificate.crt -text -noout
```

Authentication and Authorization

Introduction to Aspera Authentication and Authorization

A transfer server can use either SSH, HTTPS, or Websocket authentication and authorization for browsing and transfers.

Protocols

SSH authentication is the original method used for for authentication, and is typically used for transfers between Aspera clients and servers. SSH authentication requires a system user account that is configured with a docroot or restriction in `aspera.conf`. The user can authenticate by providing a system password or SSH key.

HTTPS authentication (using the Node API) was introduced to support browsing and transfers that are initiated through Aspera web applications (IBM Aspera Faspex, IBM Aspera Shares, and IBM Aspera on Cloud), and uses a token-based authorization security layer in addition to SSH.

Websocket authentication uses token-based authorization security, and does not use SSH.

Authorization Tokens

When the server is configured for token authorization, the server-side **ascp** process requires a valid token from the client before it can start. It is the responsibility of the client to provide this token. The Aspera web applications do this automatically through HTTPS (using the Node API). The IBM Aspera Desktop Client GUI and IBM Aspera Command-Line Interface do this automatically when connecting to Aspera web applications.

There are three types of tokens that you can use: transfer tokens, basic tokens, and bearer tokens:

- A *transfer token* authorizes specific content uploads to a destination or content downloads from a remote source. Transfer-token-based authorization is generally used for FASP transfers initiated through Aspera web applications, such as IBM Aspera Faspex, IBM Aspera Shares, and IBM Aspera Application for Microsoft SharePoint, but can be used in place of SSH authentication for other types of Aspera products. For more information, see “Transfer Token Creation (Node API)” on page 311 and “Transfer Token Generation (astokengen)” on page 313.
- A *basic token* is created from an access key ID and secret, which authorizes a transfer user access to a specific area of a storage and authenticates that user to the storage. Basic tokens are less restrictive than transfer tokens. They can be used to transfer with any Aspera server that supports access keys (all but IBM Aspera on Cloud). For more information, see “Basic Tokens” on page 323.
- A *bearer token* is created from an access key ID, access key secret, and an SSL private-public key pair. Bearer token authentication is required for transfers to and from IBM Aspera on Cloud, but can be used for transfers with all other Aspera servers, too. For more information, see “Bearer Tokens” on page 324.

Require Token Authorization: Set from the Command Line

When transfer users or groups are configured to require token authorization, only transfers initiated with a valid token (transfer token, basic token, or bearer token) are allowed to transfer to or from the server. Token authorization can be set independently for incoming transfers and outgoing transfers.

About this task

The following examples use a transfer user called `aspera_user_1`.

Procedure

1. Choose or create the transfer user on the server.

   The user should not have a password. If the system does not allow this, create a very large password.
2. Set the IBM Aspera Connect public SSH key as an authorized key for the transfer user and ensure that they own the file.

   a) Create the `.ssh` directory in the user's home folder.

   ```
   # mkdir /home/aspera_user_1/.ssh/
   ```

   b) Copy the Connect public SSH key into `.ssh` and rename it `authorized_keys` (or append the public key to `authorized_keys` if the file already exists).

   ```
   # cp /opt/aspera/var/aspera_tokenauth_id_rsa.pub /home/aspera_user_1/.ssh/authorized_keys
   ```

   c) Ensure that `.ssh` and `.ssh/authorized_keys` are owned by the user.

   ```
   # chown -R aspera_user_1:aspera_user_1 /home/aspera_user_1/.ssh
   # chmod 600 /home/aspera_user_1/.ssh/authorized_keys # chmod 700 /home/aspera_user_1

   # chmod 700 /home/aspera_user_1/.ssh
   ```
3. To require token authorization for uploads and downloads, and to set the token encryption key, run the following command:

   ```
   # asconfigurator -x
   "set_user_data;user_name,aspera_user_1;authorization_transfer_in_value,token;authorization_tr
   ansfer_out_value,token;token_encryption_key,key"
   ```

   Aspera recommends that the *key* be a random string of at least 20 characters. This command creates the following text in `aspera.conf`:

   ```
   <user>
       <name>aspera_user_1</name>
       <authorization>
           <transfer>
               <in>
                   <value>token</value>
               </in>
               <out>
                   <value>token</value>
               </out>
           </transfer>
           <token>
               <encryption_key>gj5o930t78m34ejme9dx</encryption_key>
           </token>
       </authorization>
       <file_system>
             ...
             ...
       </file_system>
   </user>
   ```

   You can also configure token-authorization settings in a <group> section to be applied to all users in the group or in the <default> section to apply them globally for all users. For instructions on how to

run **asconfigurator** commands to do so, as well as to view other token configuration options, see “User, Group and Default Configurations” on page 328.

Transfer Token Creation (Node API)

Use the the Node API to create transfer tokens.

Prerequisites:

In order to create transfer tokens with the Node API, you must set up HSTS for the Node API. For instructions, see “Node API Setup” on page 294.

For information about using the **astokengen** command-line tool for generating and decoding transfer tokens during development (for debugging purposes), see “Transfer Token Generation (astokengen)” on page 313.

Creating Transfer Tokens with Node API Calls

Curl is used to call the API, and is freely available for download for all operating systems supported by Aspera:

https://curl.haxx.se/

To generate a token, run a curl command to the `/files/upload_setup` or `/files/download_setup` endpoint (depending on what kind of token you want to generate). The request body includes a JSON object called the `transfer_requests`. The Node API output response, a `transfer_specs` JSON object, includes a token, as well as a description of who is authorized to transfer using the token, what files can be transferred, and transfer properties.

Note: When generating tokens with an IBM Aspera Shares server, the endpoints are `/node_api/files/upload_setup` and `/node_api/files/download_setup`.

Upload token

General syntax:

```
# curl -i -X POST -u node_username:node_user_password -d '{"transfer_requests" :
[{"transfer_request" : { "paths" : [{}], "destination_root" : "/" } } ] }";' http(s)://
node_server:node_port/files/upload_setup
```

This command specifies the following:

`-i` Include the HTTP header in the output.

`-X POST` Specify a POST request to the HTTP server, rather than the default GET request. (This option is not required when `-d` is used, but is included here for completeness).

`-u node_username:node_user_password` Authenticate using the Node API username and password that are associated with the transfer user who has been configured for token authorization.

`-d` Send the specified data payload to the HTTP server. The payload can be entered in the command line, as it is here, or stored in a file, as described below.

`http(s)://...` The endpoint URL.

For example, the following request allows the user, `lion`, who is associated with the Node API username, `nodeuser`, and Node API password, `nodepassword`, to upload any files from the source to any location on the destination, `serengeti.com`:

```
# curl -i -v -X POST -u nodeuser:nodepassword -d '{ "transfer_requests" :
[ { "transfer_request" : { "paths" : [{}], "destination_root" : "/" } } ] }";' http://
serengeti.com:9091/files/upload_setup
```

The response output is the following, from which you extract the token string `ATV7_HtfhDa-JwWfc6RkTwhkDUqjHeLQePiOHjIS254_LJ14_7VTA`:

```
HTTP/1.1 200 OK
Cache: no-cache
Connection: close
```

```
Content-Type: application/x-javascript
{
    "transfer_specs" : [{
        "transfer_spec" : {
                "paths" : [{}],
                "source_root" : "",
                "destination_root" : "/",
                "token" : "ATV7_HtfhDa-JwWfc6RkTwhkDUqjHeLQePiOHjIS254_LJ14_7VTA",
                "direction" : "send",
                "target_rate_cap_kbps" : 100000,
                "cipher" : "none",
                "rate_policy_allowed" : "fair",
                "rate_policy" : "fair",
                "target_rate_kbps" : 45000,
                "min_rate_kbps" : 0,
                "remote_host" : "serengti.com",
                "remote_user" : "lion",
                "ssh_port" : 22,
                "fasp_port" : 33001,
                "http_fallback" : true,
                "http_fallback_port" : 8080
        }
    }]
}
```

You can also specify the transfer request parameters in a file and refer to it in the curl command, which is particularly useful for transfer requests that list many items for source content and destination. For example, the transfer request file, `upload_setup.json`, could contain the following information for a file pair list:

```
{
   "transfer_requests" : [
      {
         "transfer_request" : {
            "destination_root" : "/",
            "paths" : [
              {
                 "destination" : "/archive/monday/texts/first_thing",
                 "source" : "/monday/first_thing.txt"
              },
              {
                 "destination" : "/archive/monday/texts/next_thing"
                 "source" : "/monday/next_thing.txt",
              },
              {
                 "destination" : "/archive/monday/texts/last_thing",
                 "source" : "/monday/last_thing.txt"
              }
           ]
         }
      }
   ]
}
```

To use this file in the curl command, specify the path to the file in the `-d` option, as follows:

```
-d @upload_setup.json
```

Download token

The method for generating a download token is the same as for an upload token, except that you use the `/files/download_setup` (or `/node_api/files/download_setup` in the case of Shares) endpoint.

Using Transfer Tokens in the Command Line

Once the token is generated, it can be used to authorize **FASP** transfers by setting the `ASPERA_SCP_TOKEN` environment variable or using the `-W` option for **ascp** (or **ascp4**) and **async** sessions.

Transfer Token Generation (astokengen)

The **astokengen** command-line tool can be used to generate and decode transfer tokens. Aspera recommends that you only use **astokengen** during development for debugging purposes, and that you use the Node API for production systems.

The Node API response includes FASP transfer parameters and the token string, whereas **astokengen** only generates only a specific type of token. See "Transfer Token Creation (Node API)" on page 311

Syntax and Options

```
# astokengen [options]
```

Option (short form)	Option (long form)	Description
-A	**--version**	Print version information.
	--mode=*mode*	Direction of the transfer mode (**send** | **recv**)
-p	**--path=***path*	Source path
	--dest=*destination*	Destination path
-u	**--user=***user*	Generate the token for this user name. This name is embedded in the token and also used to retrieve further information from `aspera.conf` (**user_value** and **token_life_seconds**).
	--source-prefix=*prefix*	Prepend the given path to each source path.
	--full-paths	Store the entire path set in the token. **Note:** This option is required when creating tokens for Ascp4 transfers.

	--file-list=*filename*	Specifies a file name that contains a list of sources for a download token. Each line of the file contains a single source and blank lines are ignored. For example:
		`/monday/first_thing.txt`
		`/monday/next_thing.txt`
		`/monday/last_thing.txt`

	--file-pair-list=*filename*	Specifies a file name that contains a multiplexed list of source and destination pairs for an upload or download token. Each pair of lines encodes one source and one destination and blank lines are ignored. For example
		`/monday/first_thing.txt`
		`/archive/monday/texts/first_thing`
		`/monday/next_thing.txt`
		`/archive/monday/texts/next_thing`
		`/monday/last_thing.txt`
		`/archive/monday/texts/last_thing`

Option (short form)	Option (long form)	Description
-v *token*		Verify token against user and path parameters.
-t *token*		Display the contents of the token.
-k *passphrase*		Passphrase to decrypt token. For use with **-t**.
-b		Assume user name and paths are encoded in base64.

General Usage Examples

- Display the contents of the token:

```
# astokengen -t token [options]
```

- Authorize uploads to a specific destination:

```
# astokengen --mode=send [options] -u user --dest=path [-v token]
```

- Authorize uploads of one or more files as source/destination pairs to a specific destination:

```
# astokengen --mode=send [options] -u user --file-pair-list=filename --dest=destination [-v
token]
```

- Authorize downloads of one or more files or directories from a specific destination:

```
# astokengen --mode=recv [options] -u user -p path [-p path …] [-v token]
```

- Authorize downloads of files specified in a file list:

```
# astokengen --mode=recv [options] -u user --file-list=filename [-v token]
```

- Authorize downloads of one or more files as source/destination pairs:

```
# astokengen --mode=recv [options] -u user --file-pair-list=filename [-v token]
```

Usage Examples

Description	Example
Common upload	In a common upload, only the destination is encoded into the token. `# astokengen --user=user --dest=path --mode=send` Source paths and file lists (**--path** and **--file-list**) are not allowed and will cause **astokengen** to fail.
Paired upload	The destination is prepended to the destinations in the paired list file and they are encoded into the token. The destinations are in the odd numbered lines of the file (1, 3, 5, 7, and so on).

Description	Example
	`# astokengen --user=user --dest=path --file-pair-list=filename --mode=send` Source paths and file lists (**--path** and **--file-list**) are not allowed and will cause **astokengen** to fail.

Common download	The specified paths are encoded into the token.
	`# astokengen --user=user --path=filepath1 --path=filepath2 --mode=recv`
	`# astokengen --user=user --file-list=filename --mode=recv`
	In this case, **--dest** and **--file-pair-list** are illegal.

Paired download	The source files from the file pair list are encoded in the token. The sources are in the even numbered lines of the file (0, 2, 4, 6, 8, etc.). `# astokengen --user=user --file-pair-list=filename --mode=recv` In this case, **--dest**, **--path** and **--file-list** are illegal.

Access Key Authentication

Access key authentication provides an alternative to entering the security credentials of a Node API user or system user. Because an access key is restricted to its own storage (local or cloud), it allows access control and usage reporting to be segregated by storage. This offers significant benefits to multi-tenant service providers and enterprise installations with multiple departments.

About this task

Access Key Support:

Access key authentication can by used by Aspera client products such as IBM Aspera Desktop Client, HSTS, HSTE, and IBM Aspera Drive. It can also be used by IBM Aspera Faspex, IBM Aspera Shares, and AIBM Aspera on Cloud transfer service. For details about using access key authentication with these products, see their documentation.

Access Key Restrictions:

- The transfer user must have a file restriction configured in `aspera.conf`, rather than a docroot. If a docroot is configured, access key creation and use fails.
- Access keys must specify the storage path. Although they can be created with no storage specified, transfers using these keys fail.

Access Key Creation:

Procedure

1. Configure the system user with a restriction and ensure that no docroot is configured:

   ```
   # asconfigurator -x "set_user_data;user_name,username;absolute,AS_NULL;file_restriction,|
   restriction"
   ```

   The format of the restriction depends on the storage type (these examples allow access to the entire storage):

Storage Type	Format Example
local storage	For Unix-like OS: • specific folder: `file:////folder/*`

Storage Type	Format Example
	• drive root: `file:////*` For Windows OS: • specific folder: `file:///c%3A/`*`folder`*`/*` • drive root: `file:///c*`
Amazon S3 and IBM Cloud Object Storage - S3	`s3://*`
Azure	`azu://*`
Azure Files	`azure-files://*`
Azure Data Lake Storage	`adl://*`
Alibaba Cloud	`oss://*`
Google Cloud	`gs://*`
HDFS	`hdfs://*`

For example, to configure the system user `xfer` with a restriction that allows full access to local storage:

```
# asconfigurator -x "set_user_data;user_name,xfer;absolute,AS_NULL;file_restriction,|
file:////*"
```

2. Assign a Node API username and password to the system user. This command requires admin permissions.

```
# /opt/aspera/bin/asnodeadmin -au node_username -p node_password -x system_user
```

For example, to assign the Node API username `nodeuser` to the system user `xfer`:

```
# /opt/aspera/bin/asnodeadmin -au nodeuser -p asperaissofast -x xfer
```

This command automatically reloads the node configuration.

3. To create access keys, send a request to the Node API `/access_keys` endpoint by using **`curl`** command.

Curl is included in many Unix-based operating systems. To determine if it is installed, run **`curl`** on the command line. If it is not installed, download it from the Curl website: https://curl.haxx.se/download.html.

To create an access key, run the following command on the server:

```
# curl -ki -u node_username:node_password -X POST https://localhost:9092/access_keys -d
@access_key_config.json
```

where *access_key_config.json* is the access key configuration file.

For example,

```
# curl -ki -u nodeadmin:superP@55wOrD -X POST https://localhost:9092/access_keys -d @/
nodeadmin/ak_client1.json
```

Access Key Configuration

The access key configuration is specified in JSON. Only the "storage" object is required; the Node API creates an access key ID and secret if they are not provided.

Note: If your access key configuration is simple, you can specify it on the command line, replacing `-d @ access_key_config.json` with an argument like `-d'{"storage": {"type":"local","path":"/projects/project1"}}'`.

```
{
    "id" : "access_key_id",
    "secret" : "access_key_secret",
    "token_verification_key" : "token_key",
    "storage" : {
         storage_configuration
         },
    "license" : {
        "customer_id" : "customer_id",
        "entitlement_id" : "entitlement_id"
    },
    "configuration" : {
        "transfer" : {
            "cipher" : "cipher",
            "policy" : "policy",
            "target_rate_kbps" : target_rate,
            "target_rate_cap_kbps" : target_rate_cap,
            "content_protection_secret" : "secret",
            "preserve_timestamps" : true|false,
            "aggressiveness" : "aggressiveness",
        },
        "server" : {
            "activity_event_logging" : true|false,
            "recursive_counts" : true|false,
            "aej_logging" : true|false
        }
    },
    "files_filelock_enabled" : true|false,
    "files_filelock_restriction" : "restriction"
    }
```

Element	Required	Type	Description
id	Optional	String	ID of the access key. Returns `209 (conflict)` if it already exists. If it is not provided, the Node API creates an ID and returns the value in the response.
secret	Optional	String	Access key secret. If it is not provided, the Node API creates a secret and returns the value in the response.
token_verification_key	Optional	String	Required when the access key is used to create a bearer token, the public key corresponding to the private key that is used to create the bearer token.
storage	**Required**	JSON	Storage specification object. See examples following this table.
license	Optional	JSON object	Entitlement information, similar to regular Aspera on Demand. This is needed when the access key logs against SafeNet.
customer_id	Optional	String	Customer ID
entitlement_id	Optional	String	ID of the entitlement
configuration	Optional	JSON object	The transfer and server configuration object.
transfer	Optional	JSON object	The transfer configuration object. Available as of 3.8.0.
cipher	Optional	String	The encryption mode and minimum cipher key length allowed by the server for transfers that are authorized by this access key. Default is unset,

Element	Required	Type	Description
			such that the transfer authorized by the access key must respect the server configuration. 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. **Note:** To ensure client compatibility when requiring encryption, use a cipher with the form `aes-XXX`, which is supported by all clients and servers. Requiring GCM causes the server to reject transfers from clients that are running a version of Ascp 3.8 or older, unless `<strict_allowed_cipher>` is set to `false`. When a client requests a shorter cipher key than is configured on the server (or in an access key that authorizes the transfer), the transfer is automatically upgraded to the server setting. For more information about how the server and client negotiate the transfer cipher, see the description of `-c` in "Ascp Command Reference" on page 121 and "Ascp4 Command Reference" on page 162. **Cipher values** • none - require unencrypted transfers (not recommended). • `aes-128`, `aes-192`, or `aes-256` - allow transfers that use an encryption cipher key that is as long or longer than the setting. These settings use the CFB or GCM mode depending on the client version and cipher requested. Supports all client versions. • `aes-128-cfb`, `aes-192-cfb`, or `aes-256-cfb` - require that transfers use the CFB encryption mode and a cipher key that is as long or longer than the setting. Supports all client versions. • `aes-128-gcm`, `aes-192-gcm`, or `aes-256-gcm` - require that transfers use the GCM encryption mode introduced in version 3.9.0 and a cipher that is as long or longer than the setting. For more information about server cipher configuration, see "aspera.conf - Authorization Configuration" on page 68.
policy	Optional	String	The policy allowed for transfers that are authorized by this access key. Value can be `high`, `regular`, `fair`, `low`, `trickle`, or `fixed`. Aspera recommends against setting the policy to `fixed`, which can result in the transfer rate

Element	Required	Type	Description
			exceeding network or storage capacity if the client also requests a high minimum transfer rate that is not capped by the server. This can decrease transfer performance and cause problems on the target storage. To avoid these problems, set the allowed policy to fair. Available as of 3.8.0.
target_rate_kbps	Optional	Integer	The default initial rate for transfers that are authorized by this access key, in kilobits per second. Available as of 3.8.0.
target_rate_cap_kbps	Optional	Integer	The maximum target rate for transfers that are authorized by this access key, in kilobits per second. Available as of 3.8.0.
content_protection_secret	Optional	String	Provide a password to require that content be encrypted by the client (enforce client-side encryption-at-rest) for transfers that are authorized by this access key. Available as of 3.8.0.
preserve_timestamps	Optional	Boolean	Set to `true` to preserve file access and modification timestamps for transfers that are authorized by this access key. The server configuration overrides the access key configuration. Timestamp support in object storage varies by provider; consult your object storage documentation to determine which settings are supported. Default is unset, such that the access key inherits the server configuration. Available as of 3.8.0.
aggressiveness	Optional	Float	The aggressiveness of transfers that are authorized by this access key in claiming available bandwidth. Value can be 0.00-1.00. Available as of 3.8.0.
server	Optional	JSON object	The server configuration object. Available as of 3.8.0.
activity_event_logging	Optional	Boolean	Set to `true` to allow the Node API to query transfers that are associated with this access key through the `/events` endpoint. The access key configuration overrides the server configuration. This option must be enabled for event reporting to IBM Aspera on Cloud. Default is unset, such that the access key inherits the server configuration. Available as of 3.8.0.
recursive_counts	Optional	Boolean	Set to `true` to enable recursive counts. The access key configuration overrides the server configuration. This option must be enabled for event reporting to IBM Aspera on Cloud. Default is unset, such that the access key inherits the server configuration. Available as of 3.8.0.
aej_logging	Optional	Boolean	Set to `true` to enable reporting to the IBM Aspera on Cloud Activity app. The access key

Element	Required	Type	Description
			configuration overrides the server configuration. This option must be enabled for activity reporting to the IBM Aspera on CloudActivity app. Default is unset, such that the access key inherits the server configuration. Available as of 3.9.0.
files_filelock_enabled	Optional	Boolean	Set to `true` to allow the access key user to create filelocks. Filelocks cannot be set if filelocks are disabled on the server (`files_filelock_enabled` is set to `false` in `aspera.conf`). Available as of 3.8.0.
files_filelock_restriction	Optional	String	Set to none to allow the access key user to write, delete, or rename files if they are not locked or if the filelock was applied by the user. Set to `write` to allow the access key user to write, delete, or rename files only if the filelock was applied by the user. Available as of 3.8.0.

Minimum Access Key Configuration - The Storage Object

The "storage" section requires different values, depending on the storage type. The following examples contain the minimum information required to create an access key, and can be cut and pasted into a text file for editing.

Local storage

```
{"storage" : {
    "type" : "local",
    "path" : "path"
}}
```

Because local storage objects are simple, you can create your access key by specifying the storage in the command line:

```
# curl -ki -u nodeadmin:superP@55wOrD -X POST https://localhost:9092/access_keys -
d'{"storage":{"type":"local","path":"/projects/project1"}}
```

Amazon S3

```
{"storage" : {
    "type" : "aws_s3",
    "endpoint" : "s3.amazonaws.com",
    "path" : "bucket/path/",
    "storage_class" : "STANDARD|REDUCED_REDUNDANCY|INFREQUENT_ACCESS",
    "server_side_encryption" : "AES256|AWS_KMS",
    "server_side_encryption_aws_kms_key_id" = "arn_encryption_key",
    "credentials" : {
        "type" : "key|iam-role|assume-role",
        "access_key_id" : "aws_access_key",
        "secret_access_key" : "secret_access_key",
        "iam_role_name" : "iam_role",
        "assume_role_arn": "arn:aws:iam::your_aws_account_id:role/role_name",
        "assume_role_external_id" : "external_id",
        "assume_role_session_name" : "session_name"
    }
}}
```

Where:

- "path" includes the bucket and file path.
- If server side encryption is set to `"AWS_KMS"`, then `"server_side_encryption_aws_kms_key_id"` is required and is set to the ARN of the encryption key (for example, `"arn:aws:kms:us-east-1:648543846928:key/er23525-8754-84g4-8sf7-4834ngigfre45"`).

- Values for credentials depend on the type of authentication you use. To authenticate with your storage access key ID and secret, only specify "access_key_id" and "secret_access_key". To authenticate with an IAM role, only specify "iam_role_name". To authenticate with an assumed IAM role, only specify "assume_role_arn", "assume_role_external_id", and "assume_role_session_name".

Azure (Block and Page Storage)

```
{"storage" : {
    "type" : "azure",
    "api" : "PAGE | BLOCK",
    "container" : "container",
    "path" : "path",
    "credentials" : {
        "storage_endpoint" : "blob.core.windows.net",
        "type": "key",
        "account" : "account_name",
        "key" : "storage_access_key"
    }
}}
```

Azure Data Lake Storage

```
"storage" : {
    "type" : "azure-datalake",
    "path" : "container/path",
    "storage_endpoint" : "data_lake_store_name.azuredatalakestore.net",
    "credentials" : {
        "type" : "ClientCredential",
        "client_id" : "client_application_id",
        "refresh_url" : "https://login.windows.net/directory_id/oauth2/token",
        "client_secret" : "secret"
    }
}
```

Azure SAS

```
{"storage" : {
    "type" : "azure_sas",
    "container" : "container",
    "path" : "path",
    "api": "BLOCK|PAGE"
    "credentials" : {
        "shared_access_signature" : "shared_url"
    }
}}
```

Where the "shared_access_signature" is the shared URL, such as `https://company.blob.core.windows.net/temp?sv=2014-02-14&sr=c&sig=yfew...79uXE%3D&st=2015-07-29T07%3A00%3A00Z&se=2018-08-06T07%3A00%3A00Z&sp=rwdl`.

Azure Files

```
{"storage" : {
    "type" : "azure-files",
    "path" : "share/path",
    "credentials" : {
        "file_service_endpoint" : "https://account.file.core.windows.net/",
        "password" : "password"
    }
}}
```

Google Cloud Storage

Authenticated by a service account with a private key:

```
{"storage": {
    "type" : "google-gcs",
    "storage_endpoint" : "storage.googleapis.com",
    "path" : "bucket/path",
    "max_segments_per_compose" : 10000,
    "credentials": {
      "type": "service_account",
      "project_id": "project_id",
      "private_key_id": "key_id",
```

```
      "private_key": "-----BEGIN PRIVATE KEY-----key_string-----END PRIVATE KEY-----\n",
      "client_email": "client_id@developer.gserviceaccount.com",
    }
}}
```

Authenticated by an OAuth token:

```
{"storage" : {
    "type" : "google-gcs",
    "storage_endpoint" : "storage.googleapis.com",
    "path" : "bucket/path",
    "max_segments_per_compose" : 1024,
    "credentials" : {
      "type" : "oauth",
      "client_id" : "client_id",
      "client_secret" : "secret"
      "project_id" : "project_id",
      "access_token" : "access_token",
      "refresh_token" : "refresh_token",
      "token_expiration" : "token_lifetime_seconds"
      "auth_uri" : "https://accounts.google.com/o/oauth2/auth",
      "token_uri" : "https://accounts.google.com/o/oauth2/token",
      "auth_provider_x509_cert_url" : "https://www.googleapis.com/oauth2/v1/certs",
      "client_x509_cert_url" : "https://www.googleapis.com/robot/v1/metadata/x509/
client_id%40developer.gserviceaccount.com"
    }
}}
```

IBM Cloud Object Storage (COS) - S3

```
{"storage" : {
  "type" : "ibm-s3",
  "path" : "bucket/path",
  "endpoint" : "s3-api.us-geo.objectstorage.service.networklayer.com",
  "credentials" : {
    "type" : "key",
    "access_key_id" : "key_id",
    "secret_access_key" : "key_secret"
  }
}}
```

4. Confirm that your access key was created and retrieve its ID by running the following command:

```
# curl -ki -u node_username:node_password -X GET https://localhost:9092/access_keys
```

The output includes the ID and configuration of all access keys. For example, the following output lists an access key is for local storage:

```
HTTP/1.1 200 OK
Cache: no-cache
Connection: close
Content-Type: application/json; charset=utf-8

[
{
    "id" : "ak_1234",
    "secret" : "j3489tht42o8y32unifhkfw38ty238h3rih",
    "token_verification_key" : "9mgr3wtl4utmf394ur2ur52jgj934864ginsrh",
    "storage" : {
        "type" : "local",
        "path" : "/"
    },
    "license" : {
        "customer_id" : "customer1",
        "entitlement_id" : "43gsdi459-23r3r-w38ron-23523ro-sr82h3r8h3r"
    },
    "configuration" : {
        "transfer" : {
            "cipher" : "aes-128",
            "policy" : "fair",
            "target_rate_kbps" : 10000,
            "target_rate_cap_kbps" : 20000,
            "content_protection_secret" : "secretsecret",
            "preserve_timestamps" : false,
            "aggressiveness" : "0.00",
        },
        "server" : {
```

```
            "activity_event_logging" : true,
            "recursive_counts" : true,
            "aej_logging" : true
        }
    },
    "files_filelock_enabled" : true,
    "files_filelock_restriction" : "none"
}
]
```

5. Test the access key.

   If your access key is configured correctly, the following command returns the files in the path that was specified in the access key configuration:

```
# curl -ki -u access_key_id:access_key_secret https://localhost:9092/files/1/files
```

Basic Tokens

A *basic token* is created from an access key ID and secret, which authorizes a transfer user access to a specific area of a storage and authenticates that user to the storage. Basic tokens are less restrictive than transfer tokens. They can be used to transfer with any Aspera server that supports access keys (all but IBM Aspera on Cloud).

About this task

Procedure

1. Create an access key for the storage and retrieve its ID and secret, as described in "Access Key Authentication" on page 315.
2. Create a basic token by encoding the `access_key_id:secret` in base64.

```
# echo -n access_key_id:access_key_secret | base64
```

   For example:

```
# echo -n diDeuFLcpG9IYdsvxj0SCq4mOohNJTKvp5Q2nRWjDgIA:aspera | base64
```

   The basic token looks similar to the following:

```
ZGlEZXVGTGNwRzlJWWRzdnhqMFNDcTRtT29oTkpUS3ZwNVEyblJXakRnSUE6YXNwZXJh
```

   If the basic token breaks across lines in the output, rerun the command using the `-w0` option to remove the line break. For example:

```
# echo -n access_key_id:access_key_secret | base64 -w0
```

3. Set the basic token as an environment variable by running the following command:

```
# export ASPERA_SCP_TOKEN="Basic token_string"
```

   You can also specify the basic token on the command line by using the `-W "Basic token_string".`
4. Transfer content.

   To upload a file, use the following syntax:

```
# ascp -i path/to/private_key_file -d source_path username@hostname:destination_path
```

   Where the path to the private key file is:

   `/opt/aspera/var/aspera_tokenauth_id_rsa`

The *destination_path* can be / to indicate the top of the access key storage, or */path* to indicate a subdirectory.

For example:

```
# ascp  -i /opt/aspera/var/aspera_tokenauth_id_rsa -d testfile03 xfer@10.0.3.4/tmp
```

Bearer Tokens

A *bearer token* is created from an access key ID, access key secret, and an SSL private-public key pair. Bearer token authentication is required for transfers to and from IBM Aspera on Cloud, but can be used for transfers with all other Aspera servers, too.

To create a bearer token with asnodeadmin, run the following command as a user with admin/root permissions. If you do not specify an SSL key file or directory, you are asked if you want to create one and the filename for the private key. The bearer token is returned in standard out.

```
# /opt/aspera/bin/asnodeadmin -u node_username -p node_user_password \
    --bearer-create \
    --access-key access_key_id \
    --user-id user_id \
    --expires-at UTC_date \
    --group-ids id1,id2,… \
    --scope-role {user|admin} \
    --token-key-length length
```

Option	Required	Type	Description
-u, --user	**Required**	String	The Node API username.
-p, --pwd, --password	**Required**	String	The Node API user's password.
--bearer-create	**Required**		
--access-key	**Required**	String	The ID of the access key that is used to create the bearer token
--user-id	**Required**	String	The ID of the user who is granted permissions to content in the storage by /permissions.
--group-ids	Optional	String	The ID of the group that is granted permissions to content in the storage by /permissions.
--expires-at	Optional	UTC time	The expiration date of the bearer token in UTC format. For example, 2016-06-23T13:21:58.453Z. Default expiration is 1 hour after token creation time.
--scope-role	Optional	String	The access level of the bearer token. Value can be **admin** (default) or **user**. **admin** can change the access key configuration, **user** cannot.
--token-key-length	Optional	Double	The length of the RSA key. Must be a power of 2 between 1024 bits (128 bytes) and 16384 bits (2048 bytes). Default key length is 4096 bits.

Asconfigurator Reference

The asconfigurator Utility

The **`asconfigurator`** utility is a command-line tool for interacting with `aspera.conf`, the file that holds most configuration settings for your Aspera transfer server.

Why Use `asconfigurator`?

Because `aspera.conf` is an XML file, users can configure their transfer server by editing the file directly. However, editing the file manually can be cumbersome and error-prone because correct syntax and structure are strictly enforced. The **`asconfigurator`** utility enables you to edit `aspera.conf` through commands and parses, validates and writes well-formed XML while also confirming that the values entered for parameters are valid.

With **`asconfigurator`**, you can edit `aspera.conf` quickly and safely, with one or two commands.

After Editing `aspera.conf`

Whether you use **`asconfigurator`** or manually edit `aspera.conf`, the file must be re-read and certain services restarted in order for the changes to take effect. For detailed information, see the *Administrator's Guide: Restarting Aspera Services* for your Aspera transfer server.

Syntax and Usage

General Syntax

```
# asconfigurator -x "command[;parameter,value;parameter,value]"
```

The ***command*** is either a **`set`** command for setting a configuration or a **`delete`** command for removing a configuration. For any ***command*** you may enter one or more set of parameters and values separated by semicolons.

Note: The user executing **`asconfigurator`** commands must meet the following requirements:

- Have write access to `aspera.conf`.
- Not be configured to use a shell that restricts command usage (**`aspshell`** does not allow the use of **`asconfigurator`**).

Commands for Setting Parameter Values

Command	Description
`set_user_data`	Sets data in the user section. For parameters and values, see "User, Group and Default Configurations" on page 328.
`set_group_data`	Sets data in the group section. For parameters and values, see "User, Group and Default Configurations" on page 328.
`set_trunk_data`	Sets data in the trunk section, which contains Vlink settings. For parameters and values, see "Trunk (Vlink) Configurations" on page 333.
`set_central_server_data`	Sets data in the central server section, which contains Aspera Central and SOAP settings. For parameters and values, see "Central Server Configurations" on page 334.
`set_database_data`	Sets data in the database section, which contains settings for use with Aspera Console (earlier than 3.0). For parameters and values, see "Database Configurations" on page 336.
`set_server_data`	Sets data in the server section, which contains transfer server feature settings for use with the Node API. For parameters and values, see "Server Configurations" on page 337.
`set_http_server_data`	Sets data in the HTTP fallback server section. For parameters and values, see "HTTP Server Configurations" on page 335.

Command	Description
set_client_data	Sets data from the client section, which holds client transfer settings. For parameters and values, see “Client Configurations” on page 340.
set_node_data	Sets data in the default section, which holds the "global" node settings. For parameters and values, see “User, Group and Default Configurations” on page 328.

Note: To reset a parameter to its default value, you can use a **set** command for the parameter with a value of **AS_NULL**.

Commands for Deleting Configurations

Delete commands can be used for removing a user, group or Vlink configuration.

Command	Description
delete_user	Deletes a user's configurations.
delete_group	Deletes a group's configurations.
delete_trunk	Deletes a Vlink's configurations.

Modifying Files other than `aspera.conf`

The general syntax above modifies the default `aspera.conf`. You can also run **asconfigurator** to modify an XML file of your choice instead of `aspera.conf`.

The command below takes a path to a file to modify. If the file does not exist, it is created.

```
# asconfigurator -x "command[;parameter,value;parameter,value]" /path/to/file
```

The command below takes paths to two files. The first file is used as a base, and the modifications are written to the second file.

```
# asconfigurator -x "command[;parameter,value;parameter,value]" /path/to/file /path/to/file1
```

Using Fitness Rules

Fitness rules allow you to apply configuration settings conditionally when specified rules are met. Fitness rules are added to `aspera.conf` configurations as attributes within XML tags, such as the following:

```
<value fitness="peer_ip"(192.168.15.81)>allow</value>
```

In the example above, the parameter is set to `allow` if the peer IP address is 192.168.15.81.

Fitness Rule Syntax:

```
# asconfigurator -x "command;parameter,value,fitness,fitness_rule(fitness_template)"
```

Fitness Rule	Example	Description
cookie()	cookie(wilcard_template)	The parameter value is applied if the cookie passed from the application matches the specified template.
peer_ip()	peer_ip(ip_address/netmask)	The parameter value is applied if the IP address of the peer (the client) matches the specified IP address and optionally, its netmask.
peer_domain()	peer_domain(wilcard_template)	The parameter value is applied if the domain of the peer (the client) matches the specified template.

For example, to set a **peer_ip** fitness rule on the **authorization_transfer_in_value** configuration so that incoming transfers from 192.168.16.70 are denied, run the following command:

```
# asconfigurator -x
"set_node_data;authorization_transfer_in_value,deny,fitness,peer_ip(192.168.16.70)"
```

Examples

Below are some example commands and usage tips.

Note: You can also see sample commands for nearly all configurations by running the following `asuser` command:

```
# /opt/aspera/bin/asuserdata -+
```

- Setting the docroot of your transfer user

  ```
  # asconfigurator -x "set_user_data;user_name,transferuser;absolute,/path/to/docroot"
  ```

- Enabling HTTP Fallback using HTTPS on port 8444.

  ```
  # asconfigurator -x "set_http_server_data;enable_https,true"
  # asconfigurator -x "set_http_server_data;https_port,8444"
  ```

  Note: You can also chain two or more parameters to set within the same command. The two commands above can be combined as follows (separated by semi-colons):

  ```
  # asconfigurator -x "set_http_server_data;enable_https,true;https_port,8444"
  ```

- Setting the global inbound target transfer rate to 80Mb/s

  ```
  # asconfigurator -x "set_node_data;transfer_in_bandwidth_flow_target_rate_default,80000"
  ```

- Getting all the configurations set on the group aspera_group

  ```
  # /opt/aspera/bin/asuserdata -g aspera_group
  ```

- Creating and enabling a Vlink with an ID of 101 and a capacity of 100Mb/s

  ```
  # asconfigurator -x "set_trunk_data;id,101;trunk_on,true;trunk_capacity,100000"
  ```

- Allowing only encrypted transfers

  ```
  # asconfigurator -x "set_node_data;transfer_encryption_allowed_cipher,aes-128"
  ```

- Setting the hostname of the Aspera server to example.com

  ```
  # asconfigurator -x "set_server_data;server_name,example.com"
  ```

- Setting the global token life back to the default value of 24 hours (86400 seconds)

  Note: You can reset any setting to its default value by setting it to AS_NULL

  ```
  # asconfigurator -x "set_node_data;token_life_seconds,AS_NULL"
  ```

Reading Output

The output for **asconfigurator** commands are structured and display feedback about the success or failure of each command.

Set commands

When successful, set commands print **success** to standard out:

```
# asconfigurator -x "set_server_data;enable_http,true"
success
```

When unsuccessful, set commands print **failure** to standard out, and an explanation of why they failed:

```
# asconfigurator -x "set_server_data;enable_http,true"
failure
Syntax Error: Syntax error.  Valid values are "assert_current","server" or"option_mask", got
"enable_htt"
```

Reading `aspera.conf` configuration settings with `asuserdata`

You can view the current configuration settings by section and all the possible parameters with their default values and corresponding **asconfigurator** syntax by running **asuserdata**.

```
# /opt/aspera/bin/asuserdata [options] [commands]
```

The **asuserdata** command must be run either from within the Aspera `bin` directory, or with the full path in front of it.

Multiple command flags can be specified per call. The option flags modify the output of command flags that follow them (but not command flags that precede them).

Command Flags

Command Flag	Description
`-u` *user*	Outputs configurations set in the user section for the specified user.
`-g` *group*	Outputs configurations set in the group section for the specified group.
-d	Outputs configurations set in the database section.
-c	Outputs configurations set in the central server section.
-t	Outputs configurations set in the HTTP server section.
-a	Outputs configurations set in all sections except the user and group section.
-s	Outputs the default specification for `aspera.conf` configurations. Similar to **-+** but does not show **asconfigurator** commands.
-+	Outputs the default specification for `aspera.conf` configurations and corresponding **asconfigurator** commands for each parameter.

Option Flags

Option Flag	Description
`-x`	Formats output as XML.
`-b`	Formats output in human readable language.

Note: To see all **asuserdata** command options, run `asuserdata -h`.

User, Group and Default Configurations

General Syntax

This collection of commands configures settings for transfer authorization, bandwidth, and encryption. These settings can apply to particular users, users in particular groups, or globally to all users.

The syntax of set commands for users, groups and global settings are:

```
# asconfigurator -x "set_user_data;user_name,username;parameter,value"
# asconfigurator -x "set_group_data;group_name,groupname;parameter,value"
# asconfigurator -x "set_node_data;parameter,value"
```

Setting or getting user/group data requires you to specify the username or group name as the first parameter of the **asconfigurator** command.

Note: Not all available parameters are listed below, only the most commonly used. To view a complete list, run the following command:

```
# /opt/aspera/bin/asuserdata -+
```

Transfer Authorizations

absolute
The docroot path of a user.

Values: (String)

authorization_transfer_in_value
Incoming transfer authorization. The **token** value only allows transfers initiated with valid tokens.

Values: `allow` (default), `deny`, `token`

authorization_transfer_out_value
Outgoing transfer authorization. The **token** value only allows transfers initiated with valid tokens.

Values: `allow` (default), `deny`, `token`

authorization_transfer_in_external_provider_url
The URL of the external authorization provider for incoming transfers.

Values: (String)

authorization_transfer_out_external_provider_url
The URL of the external authorization provider for outgoing transfers.

Values: (String)

authorization_transfer_in_external_provider_soap_action
The SOAP action required by the external authorization provider for incoming transfers.

Values: (String)

authorization_transfer_out_external_provider_soap_action
The SOAP action required by the external authorization provider for outgoing transfers.

Values: (String)

token_encryption_type
The cipher used to generate encrypted authorization tokens.

Values: `aes-128` (default), `aes-192`, `aes-256`

token_encryption_key
The secret passphrase used to generate encrypted authorization tokens. Use instead of **token_encryption_keyfile**.

Values: (String)

token_life_seconds
The length of time a token is valid in seconds. The default value is 86400 seconds (24 hours).

Values: (Number)

Transfer Bandwidth Policies

transfer_in_bandwidth_aggregate_trunk_id
The ID of the Vlink to apply to incoming transfers. A value of 0 disables the Vlink.

Values: (Number 0-255)

transfer_out_bandwidth_aggregate_trunk_id
The ID of the Vlink to apply to outgoing transfers. A value of 0 disables the Vlink.

Values: (Number 0-255)

transfer_in_bandwidth_flow_target_rate_cap
The maximum value to which the target rate for incoming transfers can be set.

Values: (Number)

transfer_out_bandwidth_flow_target_rate_cap
The maximum value to which the target rate for outgoing transfers can be set (in Kbps).

Values: (Number)

transfer_in_bandwidth_flow_target_rate_default
The default value to which the target rate for incoming transfers is set (in Kbps).

Values: (Number)

transfer_out_bandwidth_flow_target_rate_default
The default value to which the target rate for outgoing transfers is set (in Kbps).

Values: (Number)

transfer_in_bandwidth_flow_target_rate_lock
A value of false allows users to adjust the transfer rate for incoming transfers. A value of true prevents users from adjusting the transfer rate for incoming transfers.

Values: `false` (default), `true`

transfer_out_bandwidth_flow_target_rate_lock
A value of false allows users to adjust the transfer rate for outgoing transfers. A value of true prevents users from adjusting the transfer rate for outgoing transfers.

Values: `false` (default), `true`

transfer_in_bandwidth_flow_min_rate_cap
The maximum value to which the minimum rate for incoming transfers can be set (in Kbps). Transfers cannot go slower than the minimum rate.

Values: (Number)

transfer_out_bandwidth_flow_min_rate_cap
The maximum value to which the minimum rate for outgoing transfers can be set (in Kbps). Transfers cannot go slower than the minimum rate.

Values: (Number)

transfer_in_bandwidth_flow_min_rate_default
The default value to which the minimum rate for incoming transfers is set (in Kbps). Transfers cannot go slower than the minimum rate.

Values: (Number)

transfer_out_bandwidth_flow_min_rate_default
The default value to which the minimum rate for outgoing transfers is set (in Kbps). Transfers cannot go slower than the minimum rate.

Values: (Number)

transfer_in_bandwidth_flow_min_rate_lock
A value of false allows users to adjust the minimum rate for incoming transfers. A value of true prevents users from adjusting the minimum rate for incoming transfers.

Values: `false` (default), `true`

transfer_out_bandwidth_flow_min_rate_lock
A value of false allows users to adjust the minimum rate for outgoing transfers. A value of true prevents users from adjusting the minimum rate for outgoing transfers.

Values: `false` (default), `true`

transfer_in_bandwidth_flow_policy_default
The default bandwidth policy for incoming transfers. The bandwidth policy determines how transfers adjust their rates according to network conditions.

Values: `fair` (default), `fixed`, `high`, `low`

transfer_out_bandwidth_flow_policy_default
The default bandwidth policy for outgoing transfers. The bandwidth policy determines how transfers adjust their rates according to network conditions.

Values: `fair` (default), `fixed`, `high`, `low`

transfer_in_bandwidth_flow_policy_lock
A value of false allows users to adjust the bandwidth policy for incoming transfers. A value of true prevents users from adjusting the bandwidth policy for incoming transfers.

Values: `false` (default), `true`

transfer_out_bandwidth_flow_policy_lock
A value of false allows users to adjust the bandwidth policy for outgoing transfers. A value of true prevents users from adjusting the bandwidth policy for outgoing transfers.

Values: `false` (default), `true`

transfer_in_bandwidth_flow_policy_allowed
The allowed bandwidth policies for incoming transfers. The chosen value and any policy less aggressive will be allowed. In order from most to least aggressive the policies are fixed, high, fair and low.

Values: any (default), `high`, `fair`, `low`

transfer_out_bandwidth_flow_policy_allowed
The allowed bandwidth policies for outgoing transfers. The chosen value and any policy less aggressive will be allowed. In order from most to least aggressive the policies are fixed, high, fair and low.

Values: any (default), `high`, `fair`, `low`

Transfer Encryption

transfer_encryption_allowed_cipher
The type of transfer encryption accepted. When set to 'any' both encrypted and unencrypted transfers are allowed.

Values: any (default), `aes-128`, `aes-192`, `aes-256`, `none`

transfer_encryption_fips_mode
Whether transfers should be encrypted with a FIPS 140-2 certified encryption module.

Values: `false` (default), `true`

content_protection_required
Whether transferred content should be left encrypted at the destination.

Values: `false` (default), `true`

content_protection_strong_pass_required
Whether a strong passphrase is required for content protection (6 characters long, at least one letter, number and special symbol).

Values: `false` (default), `true`

Transfer File System Options

resume_suffix
The extension of files used to store metadata and enable resumption of partially completed transfers. Include a '.' in the suffix, such as: `.aspera`

Values: (String), default `.aspx`

preserve_attributes
The file creation policy. When set to none the timestamps of source files are not preserved. When set to `times` the timestamps of source files are preserved at the destination.

Values: `use client setting` (default), `none`, `times`

overwrite
Whether Aspera clients are allowed to overwrite existing files on the server.

Values: `allow` (default), `deny`

file_manifest
A file manifest is a file containing a list of everything transferred in a given transfer session. When set to `text` file manifests are generated.

Values: none (default), `text`, `disable`

file_manifest_path
The location (path) where file manifests are created.

Values: (Absolute path)

pre_calculate_job_size
The policy of calculating total job size before a transfer. If set to any, the client configuration is followed. If set to no, job size calculation is disabled before transferring.

Values: any (default), no, `yes`

replace_illegal_chars
Convert restricted Windows characters in file and directory names to a non-reserved character of your choice.

Values: (Non-reserved character)

file_filters
Exclude and include files or directories with the specified pattern in the transfer. Each entry starts with a separator, preferably "|". Add multiple entries for more inclusion and exclusion patterns. To specify an exclusion, add '- ' (- and whitespace) at the beginning of the pattern, such as `|- *2016*`. To specify an inclusion, add '+ ' (+ and whitespace) at the beginning of the pattern, such as `|+ *.jpg`.

Two symbols can be used in the setting of patterns:

* (Asterisk) Represents zero to many characters in a string, for example, *.tmp matches .tmp and abcde.tmp.

? (Question Mark) Represents one character, for example, t?p matches tmp but not temp.

Specify multiple filters as a delimited list: `|+ *.jpg|- 2016*`.

Values: (String)

partial_file_suffix
Extension to be added to the names of files that are currently only partially transferred. Include a '.' in the suffix, such as: `.aspera`

Values: (String)

file_checksum
Type of checksum to compute while reading a file. Checksums are used to verify that file contents on the destination match what was read on the destination.

Values: any (default), `md5`, `sha1`, `sha256`, `sha384`, or `sha512`

async_enabled
Whether **async** is enabled on the server.

Values: `true` (default), `false`

async_connection_timeout
The time period **async** waits to establish a connection, in seconds.

Values: (Number)

async_session_timeout
The time period **async** waits for an unresponsive session, in seconds.

Values: (Number)

Document Root Options

absolute
The absolute path of the document root (docroot), which is the area of the file system that is accessible by Aspera users.

Values: (Absolute path)

read_allowed
Whether users are allowed to transfer files from the docroot (in other words, download from the docroot).

Values: `true` (default), `false`

write_allowed
Whether users are allowed to transfer files to the docroot (in other words, upload to the docroot).

Values: `true` (default), `false`

dir_allowed
Whether users are allowed to browse files in the docroot.

Values: `true` (default), `false`

file_restriction
Restrict the files that are allowed for transfers. Restrictions are set as wildcard templates. The first character is a separator (preferably a "|") which can be used to set multiple restrictions. Restrictions are processed in order and according to the following rules:

- If a restriction starts with a "!", any files that match the rest of the wildcard template are rejected.
- If a restriction does not start with a "!", then any file that matches is allowed
- Any other files are rejected

For example: |/home/aspera/*|home/janedoe/*

Values: (Character separator)(Wildcard template)[(Character separator)(Wildcard template)]

Trunk (Vlink) Configurations

General Syntax

This collection of commands configures settings related to Vlinks, which are aggregate bandwidth caps applied to transfer sessions.

The syntax for setting trunk configurations is the following :

```
# asconfigurator -x "set_trunk_data;id,trunk_id;parameter,value"
```

Setting or getting trunk data requires you to specify the ID number of the Vlink as the first parameter of the asconfigurator command.

Note: Not all available parameters are listed below, only the most commonly used. To view a complete list, run the following command:

```
# /opt/aspera/bin/asuserdata -+
```

Vlink Configurations

trunk_id
The ID of the Vlink.

Values: (Number 1-255)

trunk_on
Whether the Vlink is enabled (**true**) or disabled (**false**).

Values: **true**, **false**

trunk_capacity
The bandwidth capacity of the Vlink (in Kbps).

Values: (Number)

Central Server Configurations

General Syntax

This collection of commands configures settings related to Aspera Central, which is a service that manages transfer server SOAP features and historical transfer data.

The syntax for setting central server parameters is the following:

```
# asconfigurator -x "set_central_server_data;parameter,value"
```

Note: Not all available parameters are listed below, only the most commonly used. To view a complete list, run the following command:

```
# /opt/aspera/bin/asuserdata -+
```

Central Server Configurations

address
The network interface address on which the Aspera Central listens. The default 127.0.0.1 enables the transfer server to accept transfer requests from the local computer. Setting the value to 0.0.0.0 allows the transfer server to accept transfer requests on all network interfaces.

Values: (Network interface address, default 127.0.0.1)

port
The port on which the Aspera Central service listens.

Values: (Number 1-65535, default 40001)

persistent_store
Whether to store transfer history locally. This should be enabled if the transfer server will be used with Faspex or Shares.

Values: `enable` (default), `disable`

persistent_store_max_age
The time in seconds to retain local transfer history data.

Values: (Number, default 86400)

persistent_store_on_error
Whether the Central server should terminate (`exit`) when an error occurs while writing to the local transfer history database, or ignore the error.

Values: `ignore` (default), `exit`

compact_on_startup
Whether to compact the local transfer history database on startup (note that this may take awhile).

Values: `ignore` (default), `exit`

files_per_session
The number of file names to be recorded for any transfer session. For example, if the value is set to 50 the first 50 filenames will be recorded for any session. A setting of 0 logs all filenames. The session will still record the number of files transferred, and the number of files completed, failed or skipped.

Values: (Number, default 1000000)

ignore_empty_files
Whether to block the logging of zero byte files (`true`) or not (`false`).

Values: `true` (default), `false`

ignore_skipped_files
Whether to block the logging of skipped files (`true`) or not (`false`).

Values: `true` (default), `false`

ignore_no_transfer_files
Whether to block the logging of files that were not transferred because they already exist at the destination (`true`) or not (`false`).

Values: `true` (default), `false`

HTTP Server Configurations

General Syntax

This collection of commands configures settings related to the Aspera HTTP server, which enables the HTTP Fallback feature.

The syntax for setting HTTP server parameters is the following :

```
# asconfigurator -x "set_http_server_data;parameter,value"
```

Note: Not all available parameters are listed below, only the most commonly used. To view a complete list, run the following command:

```
# /opt/aspera/bin/asuserdata -+
```

HTTP Server Configurations

cert_file
The absolute path to an SSL certificate file to use for HTTP Fallback. If left blank the default certificate that came with your transfer server installation will be used.

Values: (Absolute path)

key_file
The absolute path to an SSL key file to use for HTTP Fallback. If left blank the default key file that came with your transfer server installation will be used.

Values: (Absolute path)

bind_address
The network interface on which the HTTP Fallback server listens. The default value 0.0.0.0 allows the HTTP Fallback server to accept transfer requests on all network interfaces.

Values: (Network interface address, default 0.0.0.0)

restartable_transfers
Whether interrupted transfers should resume at the point of interruption (`true`) or not (`false`).

Values: `true` (default), `false`

session_activity_timeout
The amount of time in seconds that the HTTP Fallback server will wait before canceling a transfer session that can't communicate with the client. A value of 0 means the HTTP Fallback server will never timeout due to lack of communication from the client.

Values: (Number, default 20])

http_port
The port on which the HTTP server listens.

Values: (Number 1-65535, default 8080)

https_port
The port on which the HTTPS server listens.

Values: (Number 1-65535, default 8443)

enable_http
Whether HTTP Fallback is enabled for failed UDP transfers to continue over HTTP (`true`) or not (`false`).

Values: `true` (default), `false`

enable_https
Whether HTTP Fallback is enabled for failed UDP transfers to continue over HTTPS (`true`) or not (`false`).

Values: `true` (default), `false`

Database Configurations

General Syntax

This collection of commands configures settings related to the MySQL database that stores transfer data (for use with Aspera Console before version 3.0).

The syntax for setting database parameters is the following:

```
# asconfigurator -x "set_database_data;parameter,value"
```

Database Configurations

server
The IP address of the database server (or the IP address of the Aspera Console server).

Values: (IP address, default 127.0.0.1)

port
The port that the database server listens on. The default value for an Aspera Console installation is 4406.

Values: (Number 1-65535, default 4406)

user
The user login for the database server.

Values: (String)

password
The password for the database server.

Values: (String)

database_name
The name of the database used to store Aspera transfer data.

Values: (String)

threads
The number of parallel connections used for database logging.

Values: (Number, default 1)

exit_on_database_error
Whether all transfers are stopped on a database error (`true`) or not (`false`).

Values: `false` (default), `true`

session_progress
Whether transfer status should be logged at a given interval (`true`) or not (`false`). Transfer status includes number of files transferred, bytes transferred, among other stats.

Values: `true` (default), `false`

`session_progress_interval`
The frequency at which an Aspera node logs transfer session data, in seconds.

Values: (Number 1-65535, default 1)

`file_events`
Whether complete file paths and file names should be logged (`true`) or not (`false`). Performance may be impacted when setting this to true for transfers of thousands of files.

Values: `true` (default), `false`

`file_progress`
Whether file status, such as bytes transferred, should be logged (`true`) or not (`false`).

Values: `true` (default), `false`

`file_progress_interval`
The frequency with which an Aspera node logs file transfer data, in seconds.

Values: (Number 1-65535, default 1)

`files_per_session`
The number of file names to be recorded for any transfer session. For example, if the value is set to 50 the first 50 filenames will be recorded for any session. A setting of 0 logs all filenames. The session will still record the number of files transferred, and the number of files completed, failed or skipped.

Values: (Number, default 0)

`file_progress_interval`
The frequency at which an Aspera node logs file transfer data, in seconds.

Values: (Number 1-65535, default 1)

`ignore_empty_files`
Whether to block the logging of zero byte files (`true`) or not (`false`).

Values: `false` (default), `true`

`ignore_skipped_files`
Whether to block the logging of skipped files (`true`) or not (`false`).

Values: `false` (default), `true`

`ignore_no_transfer_files`
Whether to block the logging of files that were not transferred because they already exist at the destination (`true`) or not (`false`).

Values: `false` (default), `true`

Server Configurations

General Syntax

This collection of commands configures settings related to transfer server features such as the Aspera Node API service (**asperanoded**), Aspera Watch Service, Aspera Watchfolders, and Aspera Proxy.

The syntax for setting server parameters is the following:

```
# asconfigurator -x "set_server_data;parameter,value"
```

Note: Not all available parameters are listed below, only the most commonly used. To view a complete list, run the following command:

```
# /opt/aspera/bin/asuserdata -+
```

Transfer Server

server_name
The hostname or IP address of this Aspera transfer server.

Values: (String)

transfers_multi_session_default
The default value for the number of sessions in a multi-session transfer.

Values: (Number, default 1)

transfers_retry_duration
The time duration during which transfer retries are attempted.

Values: (Time value, default 20m)

transfers_retry_all_failures
Whether a transfer should be retried after all failures (`true`) or not (`false`). If set to false, transfers won't be retried for failured deemed unretryable, such as for permission failures.

Values: `false` (default), `true`

http_port
The HTTP port on which the **asperanoded** service listens.

Values: 9091 (default). Must be above 1024.

https_port
The HTTPS port on which the **asperanoded** service listens.

Values: 9092 (default). Must be above 1024.

enable_http
Whether HTTP is enabled for **asperanoded** on the port configured for **http_port** (`true`) or not (`false`).

Values: `false` (default), `true`

enable_https
Whether HTTPS is enabled for **asperanoded** on the port configured for **https_port** (`true`) or not (`false`).

Values: `true` (default), `false`

cert_file
The full path of the SSL certificate file for **asperanoded**.

Values: (Absolute file path)

ssh_host_key_fingerprint
The SSH key fingerprint used by Aspera clients to determine the server's authenticity. The client confirms a server's authenticity by comparing the server's fingerprint with the trusted fingerprint.

Values: (String)

ssh_host_key_path
The path to the transfer server's public or private key file, from which the fingerprint is extracted automatically.

Values: (Absolute file path)

ssh_port
The port to use for SSH authentication of transfer users.

Values: (Number, default 33001)

max_response entries
The maximum number of items the Node API will return on calls.

Values: (Number, default 1000)

max_response time_sec
The time limit in seconds before an unresponsive Node API response times out.

Values: (Number, default 10)

db_dir
The path to the directory where the redis database file for the Node API is saved.

Values: (Absolute path)

db_port
The port on which the redis database for the Node API listens.

Values: (Number, default 31415)

activity_logging
Whether transfer logs should be queriable via the Node API (`true`) or not (`false`).

Values: `false` (default), `true`

watchd_enabled
Whether the Watchfolder (**asperawatchd**) service is enabled (`true`) or not (`false`).

Values: `false` (default), `true`

ssl_ciphers
The list of SSL encryption ciphers that the server will allow. Each cipher is separated by a colon (:). See the server documentation for the default list of ciphers.

Values: (Colon-delimited list)

ssl_protocol
The minimum allowed SSL protocol. Higher security protocols are always allowed.

`tlsv1` (default), `tlsv1.1`, `tlsv1.2`

Aspera Proxy

proxy_enabled
Whether forward proxy is on (`true`) or off (`false`).

Values: `false` (default), `true`

proxy_authentication
Whether to enable the authentication requirement for the forward proxy server (`true`) or not (`false`).

Values: `false` (default), `true`

proxy_bind_ip_address
The IP address that the forward proxy server binds to (also the IP address that the client connects to). 0.0.0.0 allows the proxy server to bind to all available interfaces.

Values: (IP address, default 0.0.0.0)

proxy_bind_ip_netmask
The netmask that the forward proxy server binds to (also the netmask that the client connects to).

Values: (String)

proxy_port_range_low
The lower bound of the port range for the forward proxy.

Values: (Number, default 5000])

proxy_port_range_high
The upper bound of the port range for the forward proxy.

Values: (Number, default 10000)

proxy_cleanup_interval
The interval in seconds at which the forward proxy server scans and cleans up expired sessions.

Values: (Number, default 0)

proxy_keepalive_internal
The interval in seconds at which the ascp client sends keep-alive requests. This option is propogated to the client.

Values: (Number, default 0)

proxy_session_timeout
The interval in seconds after which a session times out if no keep-alive updates have been received.

Values: (Number, default 0)

rproxy_rules_rule_proxy_port
The reverse proxy server port that receives UDP traffic.

Values: (Number, default 33001)

rproxy_rules_rule_host
The IP address and SSH port of the internal destination. If unspecified the default port is 22.

Values: (IP address and port)

rproxy_rules_rule_hosts
The list of IP addresses and SSH ports for the load-balancing feature. The first character is a separator (preferably a "|") which can be used to set multiple hosts. For example: |10.0.23.123:33001|10.0.23.124:33001|10.0.23.125:33001

Values: (Character separator)(IP address)[(Character separator)(IP address)]

rproxy_rules_rule_squash_user
The account name used for authenticating with the internal server.

Values: (String)

rproxy_rules_rule_key_file
The path to the SSH private key for authenticating with the internal server.

Values: (Absolute path)

rproxy_rules_rule_udp_port_reuse
Whether the reverse proxy should reuse the UDP port (`true`) or not (`false`). Setting this to false enables reverse proxy to create iptables rules that increment the UDP port number that clients connect to, and the internal server's UDP port to which transfers are routed to.

Values: `true` (default), `false`

rproxy_rules_rule_balancing
The method for distributing transfers as part of the load balancing feature. Currently `round-robin` is the only supported method.

Values: `round-robin` (default)

rproxy_enabled
Whether reverse proxy is on (true) or off (false).

Values: `false` (default), `true`

rproxy_log_level
The level of debug messages to log for reverse proxy.

Values: `0` (default), 1, 2

rproxy_log_directory
The reverse proxy server log file location. If no value is set, the proxy logs to syslog.

Values: (Absolute path)

Client Configurations

General Syntax Guidelines

This collection of commands configures settings related to client transfers, which are transfers you initiate with **ascp** on the command line or the GUI of your product.

The syntax for setting client parameters is the following:

```
# asconfigurator -x "set_client_data;parameter,value"
```

Note: Not all available parameters are listed below, only the most commonly used. To view a complete list, run the following command:

```
# /opt/aspera/bin/asuserdata -+
```

Parameters and Values

`transport_cipher`
The encryption cipher to use for transfers.

Values: `aes-128` (default), `aes-192`, `aes-256`, `none`

`ssl_ciphers`
The list of SSL encryption ciphers that the server will allow. Each cipher is separated by a colon (:). See the server documentation for the default list of ciphers.

Values: (Colon-delimited list)

`ssl_protocol`
The minimum allowed SSL protocol. Higher security protocols are always allowed.

Values: `tlsv1` (default), `tlsv1.1`, `tlsv1.2`

`default_ssh_key`
The path to the default SSH key that should be used in command line transfers.

Values: (Absolute path)

Troubleshooting

Solutions to common problems.

Clients Cannot Establish Connection

Learn how to troubleshoot client issues with connecting to HSTS.

Procedure

1. Test SSH ports and HTTP/HTTPS ports.
   a) On the client computer, run the following command:

   ```
   # telnet server_ip_address port
   ```

   For example, to test connection to 10.0.1.1 through TCP/33001, you run the following command:

   ```
   # telnet 10.0.1.1 33001
   ```

   b) If the client cannot establish connections to the ports, verify the port number and the firewall configuration of HSTS. Also make sure that the client firewall allows outbound connections.
2. Test UDP ports.

   If you can establish an SSH connection but not run a FASP file transfer, there might be a firewall blockage of FASP's UDP port.
3. Verify SSH service status

   If there is no firewall blockage between the client and HSTS, on the client machine, try establishing a SSH connection: (HSTS address: 10.0.1.1, TCP/33001)

   ```
   # ssh aspera_user_1@10.0.1.1 -p 33001
   ```

   If the SSH service runs normally, the client should see a message prompting to continue the connection or for a password. However, if you see a "Connection Refused" message, which indicates that the SSH service isn't running, review your SSH service status. Ignore the "permission denied" message after entering the password, which is discussed in next steps.

4. Applied authentication method is enabled in SSH

   If you can establish a SSH connection, but it returns "permission denied" message, the SSH Server on HSTS might have password authentication disabled:

   ```
   Permission denied (publickey,keyboard-interactive).
   ```

   Open your SSH Server configuration file with a text editor:

   ```
   /etc/ssh/sshd_config
   ```

   To allow public key authentication, add or uncomment the *PubkeyAuthentication yes*. To allow password authentication, add or uncomment *PasswordAuthentication yes*. Here is a configuration example:

   ```
   ...
   PubkeyAuthentication yes
   PasswordAuthentication yes
   ...
   ```

   To activate your changes, restart the SSH server.
5. Restart the SSH server to apply new settings.

   Restarting your SSH server does not affect currently connected users.

   ```
   # systemctl restart sshd.service
   ```

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

   ```
   # service sshd restart
   ```
6. Verify that the user credentials are correct, and the user has sufficient access permissions to their docroot

   a) Attempt to establish an SSH connection:

   ```
   # ssh username@server_ip_address -p  port
   ```

   For example:

   ```
   $ ssh aspera_user_1@10.0.1.1 -p 33001
   ```

   b) Enter the user's password.

   If you see "Permission denied" message, you may have a wrong user credentials, or the user doesn't have sufficient access permissions to its docroot.

Error: Session Timeout During Ascp Transfers

If you attempt an Ascp transfer over a network with high latency or to/from storage with slow read/write, you might receive a timeout error message. You can increase the timeout to allow your transfers to complete.

About this task

The message is similar to the following:

```
ERR Failed to receive Close Session, read timed out (errno=110) timeout:120, rsize=0
```

To increase the timeout, follow these steps:

Procedure

1. Run the following **asconfigurator** command:

```
# asconfigurator -x "set_node_data;session_timeout_sec,time"
```

where *time* is the desired time in seconds before timeout. This creates the following text in `aspera.conf`:

```
<default>
     <session_timeout_sec>time</session_timeout_sec>
</default>
```

2. Alternatively, manually edit `aspera.conf`.

   The `aspera.conf` configuration file is in the following location:

   `/opt/aspera/etc/aspera.conf`

Node API Transfers of Many Small Files Fails

Ascp transfers that are started through the Node API or Watch Folders to or from servers that have Unix-like OS can fail when transferring many (millions) of small files because the Redis database exceeds available number of file descriptors.

To increase the maximum number of file descriptors from the default of 1024 to a larger value, such as 1,000,000, run the following command:

```
$ ulimit -Sn 10000000
```

Logs Overwritten Before Transfer Completes

The logs of long transfers of many (millions) of files can be overwritten before the session completes, potentially deleting useful troubleshooting information if an error or failure occurs. To avoid this problem, set the log size to a larger value than the default of 10 MB. For information on other logging configuration options, see “Server Logging Configuration for Ascp and Ascp4” on page 107.

About this task

Logging settings are configured by running **asconfigurator** commands (recommended) or by manually editing `aspera.conf`.

To increase log size by using asconfigurator:

Run the following command:

```
# asconfigurator -x "set_logging_data;log_size,size_mb"
```

To increase log size by manually editing `aspera.conf`:

Procedure

1. Open `aspera.conf` in a text editor run with administrator privileges.

   `/opt/aspera/etc/aspera.conf`

2. Add the <logging> section to the <default> section:

```
...
<default>
   <file_system>...</file_system>
  <logging>
      <log_size>size</log_size>
   </logging>
</default>
...
```

   Where *size* is the log size in MB.

3. Save your changes.

4. Validate the XML form of `aspera.conf`:

```
# /opt/aspera/bin/asuserdata -v
```

Disabling SELinux

SELinux (Security-Enhanced Linux), an access-control implementation, can prevent web UI access.

About this task

To disable SELinux:

Procedure

1. Open the SELinux configuration file: `/etc/selinux/config`.
2. Locate the following line:

```
SELINUX=enforcing
```

3. Change the value to **disabled**:

```
SELINUX=disabled
```

   Save your changes and close the file.
4. On the next reboot, SELinux is permanently disabled. To dynamically disable it before the reboot, run the following command:

```
# setenforce 0
```

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
```

IBM Aspera HTTPD

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

Run the following commands to restart asperahttpd:

```
# systemctl restart asperahttpd
```

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

```
# service asperahttpd restart
```

Docroot vs. File Restriction

A transfer user's access to the server's file system can be restricted by configuring a docroot or a file restriction. Though similar, certain Aspera features require that the transfer user have a file restriction rather than a docroot.

Note: A configuration (global, group, or user) can have a docroot or a file restriction; configurations with both are not supported.

	Docroot	**File Restriction**
Required for	• Server-side encryption-at-rest (docroot in URI format) • Connecting the node to IBM Aspera Faspex, IBM Aspera Shares, IBM Aspera Console, or IBM Aspera Application for Microsoft SharePoint	• Complex file-system access rules • Creating access keys with the Node API • Connecting the node to IBM Aspera on Cloud
Syntax	An absolute pathname that can include a substitutional string. Supported strings: • `$(name)` • `$(home)` The pathname can be in URI format; special characters must be URL-encoded.	A set of file system filters that use "*" as a wildcard and "!" to indicate "exclude". Paths are in URI format; special characters in a URI must be URL-encoded. Access to a file is rejected unless the file matches the restrictions, which are processed in the following order: • If a restriction starts with "!", the user is not allowed to access any files that match the rest of the restriction. • If a restriction does not start with "!", the user can access any file that matches the filter. • If one or more restrictions do not start with "!", the user can access any file that matches any one of the no-"!" restrictions.

Examples	• As an absolute path:
	`/docs`
	• With a substitutional string:
	`/users/$(name)`
	• As a URI:
	`s3://s3.amazonaws.com/my_bucket`
	or
		• For a specific folder:
		`file:////docs/*`
		• For the drive root:
		`file:///c*`
		• For ICOS-S3 storage:
		`s3://my_vault/*`
		• To exclude access to key files:
		`!*.key`

	Docroot	File Restriction
	`file:///docs`	For more examples, see “Getting Started with Watch Folders ” on page 186
How to set	See “Setting Up Transfer Users ” on page 59.	See “Getting Started with Watch Folders ” on page 186.

URL Encoding Characters

The following reserved characters are often included in passwords and secret keys:

Character	!	#	$	&	'	(	)	*	+
URL encoded	%21	%23	%24	%26	%27	%28	%29	%2A	%2B

Character	.	/	:	;	=	?	@	[	]
URL encoded	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

To URL encode other characters and to encode entire strings at once, you may use the online tool:

http://www.url-encode-decode.com/

Select **UTF-8** as the target.

Aspera Ecosystem Security Best Practices

About this task

Your Aspera applications can be configured to maximize system and content security. The following sections describe the recommended settings and practices that best protect your content when using IBM Aspera High-Speed Transfer Server and IBM aspera High-Speed Transfer Endpoint.

Contents

Securing the Systems that Run Aspera Software

Securing the Aspera Application

Securing Content in your Workflow

Securing the Systems that Run Aspera Software

About this task

The systems that run Aspera software can be secured by keeping them up to date, by applying security fixes, and by configuring them using the recommended settings.

Updates

Aspera continually improves the built-in security of its products, as do the producers of third-party components used by Aspera, such as Nginx, and OpenSSH. One of the first lines of defense is keeping your products up to date to ensure that you are using versions with the latest security upgrades:

- Keep your operating system up to date.
- Keep your Aspera products up to date.
- If using, keep OpenSSH up to date. The server security instructions require that OpenSSH 4.4 or newer (Aspera recommends 5.2 or newer) is installed on your system in order to use the `Match` directive. `Match` allows you to selectively override certain configuration options when specific criteria (based on user, group, hostname, or address) are met.

Security Fixes

Rarely, security vulnerabilities are detected in the operating systems and third-party components that are used by Aspera. Aspera publishes security bulletins immediately that describe the affected products and recommended remediation steps.

Security Configuration

Recommended security settings vary depending on the products you are using and how they interact. See the following subsections for your Aspera products.

HSTS

Procedure

1. Configure your SSH Server.

   Aspera recommends that you:

   - Open TCP/33001 and keep TCP/22 open until users are notified that they should switch to TCP/33001.
   - Once users are notified, block TCP/22 and allow traffic only on TCP/33001.

   The following steps open TCP/33001 and block TCP/22.

   a) Open the SSH configuration file.

   ```
   /etc/ssh/sshd_config
   ```

   If you do not have an existing configuration for OpenSSH, or need to update an existing one, Aspera recommends the following reference: https://wiki.mozilla.org/Security/Guidelines/OpenSSH.

   b) Change the SSH port from TCP/22 to TCP/33001.

   Add TCP/33001 and comment out TCP/22 to match the following example:

   ```
   #Port 22
   Port 33001
   ```

   HSTS admins must also update the `SshPort` value in the <`WEB...`> section of `aspera.conf`.

   Once this setting takes effect:

   - Aspera clients must set the TCP port to 33001 when creating connections in the GUI or specify **`-P 33001`** for command line transfers.
   - Server administrators should use `ssh -p 33001` to access the server through SSH.

   c) Disable non-admin SSH tunneling.

   SSH tunneling can be used to circumvent firewalls and access sensitive areas of your company's network. Add the following lines to the end of `sshd_config` (or modify them if they already exist) to disable SSH tunneling:

   ```
   AllowTcpForwarding no
   Match Group root
   AllowTcpForwarding yes
   ```

   Depending on your `sshd_config` file, you might have additional instances of `AllowTCPForwarding` that are set to the default `Yes`. Review your `sshd_config` file for other instances and disable if necessary.

   Disabling TCP forwarding does not improve security unless users are also denied shell access, because with shell access they can still install their own forwarders. Aspera recommends assigning users to aspshell, described in the following section.

   d) Disable password authentication and enable public key authentication.

Public key authentication provides a stronger authentication method than passwords, and can prevent brute-force SSH attacks if all password-based authentication methods are disabled.

Important: Before proceeding:

- Create a public key and associate it with a transfer user, otherwise clients have no way of connecting to the server.

  For instructions on using public key authentication, see “Creating SSH Keys ” on page 152 and “Setting Up a User's Public Key on the Server” on page 24.
- Configure at least one non-root, non-transfer user with a public key to use to manage the server. This is because in the following steps, root login is disabled and transfer users are restricted to aspshell, which does not allow interactive login. This user and public key is what you use to access and manage the server as an administrator.

Add or uncomment `PubkeyAuthentication yes` and comment out `PasswordAuthentication yes`:

```
PubkeyAuthentication yes
#PasswordAuthentication yes
PasswordAuthentication no
```

Note: If you choose to leave password authentication enabled, be sure to advise account creators to use strong passwords and set `PermitEmptyPasswords` to "no".

```
PermitEmptyPasswords no
```

e) Disable root login.

CAUTION: This step disables root access. Make sure that you have at least one user account with sudo privileges before continuing, otherwise you may not have access to administer your server.

Comment out `PermitRootLogin yes` and add `PermitRootLogin No`:

```
#PermitRootLogin yes
PermitRootLogin no
```

f) Restart the SSH server to apply new settings. Restarting your SSH server does not affect currently connected users.

```
isi services -a sshd disable
# isi services -a sshd enable
```

```
# systemctl restart sshd.service
```

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

```
# service sshd restart
```

g) Review your logs periodically for attacks.

For information on identifying attacks, see IBM Aspera IBM Aspera High-Speed Transfer Server Admin Guide: Securing Your SSH Server.

2. Configure your server's firewall to permit inbound access to only Aspera-required ports.

   Aspera requires inbound access on the following ports:

   - For SSH connections that are used to set up connections, TCP/33001.
   - For FASP transfers, UDP/33001.
   - If you use HTTP and HTTPS fallback with HSTS, TCP/8080 and TCP/8443. If you only use HTTPS, only open TCP/8443.
3. For HSTS, require strong TLS connections to the web server.

TLS 1.0 and TLS 1.1 are vulnerable to attack. Run the following command to require that the client's SSL security protocol be TLS version 1.2 or higher:

```
# /opt/aspera/bin/asconfigurator -x "set_server_data;ssl_protocol,tlsv1.2"
```

4. If asperanoded is exposed to internet traffic, run it behind a reverse proxy.

   If your Aspera server must expose asperanoded to the internet, such as when setting it up as a IBM Aspera on Cloud (AoC) node, Aspera strongly recommends protecting it with a reverse proxy. Normally, asperanoded runs on port 9092, but nodes that are added to AoC must have asperanoded run on port 443, the standard HTTPS port for secure browser access. Configuring a reverse proxy in front of asperanoded provides additional protection (such as against DOS attacks) and resource handling for requests to the node's 443 port.
5. Install Aspera FASP Proxy in a DMZ to isolate your HSTS from the Internet.

   For more information, see IBM Aspera FASP Proxy Admin Guide

Securing the Aspera Applications

About this task

Your Aspera products can be configured to limit the extent to which users can connect and interact with the servers. The instructions for Shares 1.9.x and Shares 2.x are slightly different; see the section for your version.

HSTS

Procedure

1. Restrict user permissions with **aspshell**.

   By default, all system users can establish a FASP connection and are only restricted by file permissions. Restrict the user's file operations by assigning them to use **aspshell**, which permits only the following operations:

   - Running Aspera uploads and downloads to or from this computer.
   - Establishing connections between Aspera clients and servers.
   - Browsing, listing, creating, renaming, or deleting contents.

   These instructions explain one way to change a user account or active directory user account so that it uses the **aspshell**; there may be other ways to do so on your system.

   Run the following command to change the user login shell to **aspshell**:

   ```
   # sudo usermod -s /bin/aspshell username
   ```

   Confirm that the user's shell updated by running the following command and looking for `/bin/aspshell` at the end of the output:

   ```
   # grep username /etc/passwd
   username:x:501:501:...:/home/username:/bin/aspshell
   ```

   Note: If you use OpenSSH, sssd, and Active Directory for authentication: To make `aspshell` the default shell for all domain users, first set up a local account for server administration because this change affects all domain users. Then open `/etc/sssd/sssd.conf` and change `default_shell` from `/bin/bash` to `/bin/aspshell`.
2. Restrict Aspera transfer users to a limited part of the server's file system or bucket in object storage.

   a) For on-premises servers, set a default docroot to an empty folder, then set a docroot for each user:

   ```
   # asconfigurator -x "set_node_data;absolute,docroot"
   # asconfigurator -x "set_user_data;user_name,username;absolute,docroot"
   ```

Replace *username* with the username and *docroot* with the directory path to which the user should have access.

b) For cloud-based servers, set a default restriction to an empty folder, then set a restriction for each user:

```
# asconfigurator -x "set_node_data;file_restriction,|storage_path"
# asconfigurator -x "set_user_data;user_name,username;file_restriction,|storage_path"
```

Replace *username* with the username and *storage_path* with the path to which the user has access. Restriction syntax is specific to the storage:

Storage Type	Format Example
local storage	`file:///*`
S3 and IBM Cloud Object Storage	`s3://*`
Swift storage	`swift//*`
Azure storage	`azu://*`
Azure Files	`azure-files://*`
Google Cloud Storage	`gs://*`
Hadoop (HDFS)	`hdfs://*`

The "|" is a delimiter, and you can add additional restrictions. For example, to restrict the system user `xfer` to `s3://s3.amazonaws.com/bucket_xyz/folder_a/*` and not allow access to key files, run the following command:

```
# asconfigurator -x "set_user_data;user_name,xfer;file_restriction,|s3://s3.amazonaws.com/
bucket_xyz/folder_a/*|!*.key"
```

3. Restrict users' read, write, and browse permissions.

   Users are given read, write, and browse permissions to their docroot by default. Change the global default to deny these permissions:

   ```
   # asconfigurator -x "set_node_data;read_allowed,false;write_allowed,false;dir_allowed,false"
   ```

   Run the following commands to enable permissions per user, as required:

   ```
   # asconfigurator -x "set_user_data;user_name,username;read_allowed,false"
   # asconfigurator -x "set_user_data;user_name,username;write_allowed,false"
   # asconfigurator -x "set_user_data;user_name,username;dir_allowed,false"
   ```

4. Limit transfer permissions to certain users.

   Set the default transfer permissions for all users to deny:

   ```
   # asconfigurator -x "set_node_data;authorization_transfer_in_value,deny"
   # asconfigurator -x "set_node_data;authorization_transfer_out_value,deny"
   ```

   Allow transfers for specific users by running the following commands for each user:

   ```
   # asconfigurator -x "set_user_data;user_name,username;authorization_transfer_in_value,allow"
   # asconfigurator -x "set_user_data;user_name,username;authorization_transfer_out_value,allow"
   ```

   Note: For a user that is used by Shares or Faspex (usually `xfer`), allow transfers only with a token by setting `authorization_transfer_{in|out}_value` to `token`.

5. Encrypt transfer authorization tokens.

   When a client requests a transfer from a server through an Aspera web application, an authorization token is generated. Set the encryption key of the token for each user or group on the server:

```
# asconfigurator -x "set_user_data;user_name,username;token_encryption_key,token_string"
# asconfigurator -x "set_group_data;group_name,groupname;token_encryption_key,token_string"
```

The token string should be at least 20 random characters.

Note: This is not used to encrypt transfer data, only the authorization token.

6. Require encryption of content in transit.

   Your server can be configured to reject transfers that are not encrypted, or that are not encrypted with a strong enough cipher. Aspera recommends setting an encryption cipher of at least AES-128. AES-192 and AES-256 are also supported but result in slower transfers. Run the following command to require encryption:

   ```
   # asconfigurator -x "set_node_data;transfer_encryption_allowed_cipher,aes-128"
   ```

   By default, your server is configured to transfer (as a client) using AES-128 encryption. If you require higher encryption, change this value by running the following command:

   ```
   # asconfigurator -x "set_client_data;transport_cipher,value"
   ```

   You can also specify the encryption level in the command line by using `-c cipher` with **ascp** and **async** transfers. **ascp4** transfers use AES-128 encryption.

7. Configure SSH fingerprinting for HSTS.

   For transfers initiated by a web application (such as Faspex, Shares, or Console), the client browser sends the transfer request to the web application server over an HTTPS connection. The web application requests a transfer token from the target server. The transfer is executed over a UDP connection directly between the client and the target server and is authorized by the transfer token. Prior to initiating the transfer, the client can verify the server's authenticity to prevent server impersonation and man-in-the-middle (MITM) attacks.

   To verify the authenticity of the transfer server, the web application passes the client a trusted SSH host key fingerprint of the transfer server. The client confirms the server's authenticity by comparing the server's fingerprint with the trusted fingerprint. In order to do this, the host key fingerprint or path must be set in the server's `aspera.conf`.

   Note: Server SSL certificate validation (HTTPS) is enforced if a fingerprint is specified in `aspera.conf` and HTTP fallback is enabled. If the transfer "falls back" to HTTP and the server has a self-signed certificate, validation fails. The client requires a properly signed certificate.

   If you set the host key path, the fingerprint is automatically extracted from the key file and you do not extract it manually.

   Retreiving and setting the host key fingerprint:

   a) Retrieve the server's SHA-1 fingerprint.

   ```
   # cat /etc/ssh/ssh_host_rsa_key.pub | awk '{print $2}' | base64 -d | sha1sum
   ```

   b) Set the SSH host key fingerprint in `aspera.conf`. (Go to the next step to set the host key path instead).

   ```
   # asconfigurator -x "set_server_data;ssh_host_key_fingerprint,fingerprint"
   ```

   This command creates a line similar to the following example of the `<server>` section of `aspera.conf`:

   ```
   <ssh_host_key_fingerprint>7qdOwebGGeDeN7Wv+2dP3HmWfP3
   </ssh_host_key_fingerprint>
   ```

   c) Restart the node service to activate your changes.

Run the following commands to restart `asperanoded`:

```
# systemctl restart asperanoded
```

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

```
# service asperanoded restart
```

```
isi services -a asperanoded disable
# isi services -a asperanoded enable
```

Setting the host key path: To set the SSH host key path instead of the fingerprint, from which the fingerprint will be extracted automatically, run the following command:

```
# asconfigurator -x "set_server_data;ssh_host_key_path,ssh_key_filepath"
```

This command creates a line similar to the following in the `<server>` section of `aspera.conf`:

```
<ssh_host_key_path>/etc/ssh/ssh_host_rsa_key.pub
</ssh_host_key_path>
```

Restart the node service to activate your changes, as described for "Retreiving and setting the host key fingerprint".

8. Install properly signed SSL certificates.

   Though your Aspera server automatically generates self-signed certificates, Aspera recommends installing valid, signed certificates. These are required for some applications.

Securing Content in your Workflow

Procedure

1. If your workflow allows, enable server-side encryption-at-rest (EAR).

   When files are uploaded from an Aspera client to the Aspera server, server-side encryption-at-rest (EAR) saves files on disk in an encrypted state. When downloaded from the server, server-side EAR first decrypts files automatically, and then the transferred files are written to the client's disk in an unencrypted state. Server-side EAR provides the following advantages:

   - It protects files against attackers who might gain access to server-side storage. This is important primarily when using NAS storage or cloud storage, where the storage can be accessed directly (and not just through the computer running HSTS).
   - It is especially suited for cases where the server is used as a temporary location, such as when one client uploads a file and another client downloads it.
   - Server-side EAR can be used together with client-side EAR. When used together, content is doubly encrypted.
   - Server-side EAR doesn't create an "envelope" as client-side EAR does. The transferred file stays the same size as the original file. The server stores the metadata necessary for server-side EAR separately in a file of the same name with the file extension `.aspera-meta`. By contrast, client-side EAR creates a envelope file containing both the encrypted contents of the file and the encryption metadata, and it also changes the name of the file by adding the file extension `.aspera-env`.)
   - It works with both regular transfers (FASP) and HTTP fallback transfers.

   Limitations and Other Considerations

   - Server-side EAR is not designed for cases where files need to move in an encrypted state between multiple computers. For that purpose, client-side EAR is more suitable: files are encrypted when they first leave the client, then stay encrypted as they move between other computers, and are decrypted when they reach the final destination and the passphrase is available. See Step 4 of this section for more information on client-side encryption.

- Do not mix server-side EAR and non-EAR files in transfers, which can happen if server-side EAR is enabled after the server is in use or if multiple users have access to the same area of the file system but have different EAR configurations. Doing so can cause problems for clients by overwriting files when downloading or uploading and corrupting metadata.
- Server-side EAR does not work with multi-session transfers (using **ascp -C** or node API `multi_session` set to greater than 1) or Watch Folders (versions prior to 3.8.0 that do not support URI docroots).

To enable server-side EAR:

a) Set users' docroots in URI format (local docroots are prepended with `file:///`).

```
# asconfigurator -x "set_user_data;user_name,username;absolute,file:///path"
```

b) Set the server-side EAR password.

Set a different EAR password for each user or group:

```
# asconfigurator -x
"set_user_data;user_name,username;transfer_encryption_content_protection_secret,passphrase"
# asconfigurator -x
"set_group_data;group_name,group_name;transfer_encryption_content_protection_secret,passphrase"
```

Important: If the EAR password is lost or `aspera.conf` is compromised, you cannot access the data on the server.

c) Require content protection and strong passwords.

These settings cause server-side EAR to fail if a password is not given or if a password is not strong enough. For example, the following **asconfigurator** command adds both these options for all users (global):

```
# asconfigurator -x "set_node_data;transfer_encryption_content_protection_required,true"
# asconfigurator -x "set_node_data;transfer_encryption_content_protection_strong_pass_required,true"
```

2. Never use "shared" user accounts.

Configure each user as their own Aspera transfer user. Sharing Aspera transfer user account credentials with multiple users limits user accountability (you cannot determine which of the users sharing the account performed an action).

3. Use passphrase-protected private keys.

The **ssh-keygen** tool can protect an existing key or create a new key that is passphrase protected.

If you cannot use private key authentication and use password authentication, use strong passwords and change them periodically.

4. If your workflow allows, require client-side encryption-at-rest (EAR).

Aspera clients can set their transfers to encrypt content in transit and on the server, and the server can be configured to require client-side EAR. You can combine client-side and server-side EAR, in which case files are doubly encrypted on the server. Client-side encryption-at-rest is not supported for **ascp4** or **async** transfers.

Client configuration

The client specifies a 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. Users can enable client-side EAR in the GUI or on the **ascp** command line.

GUI: Go to **Connections > *connection_name* > Security**. Select **Encrypt uploaded files with a password** and set the password. Select **Decrypt password-protected files downloaded** and enter the password.

Ascp command line: Set the encryption and decryption password as the environment variable ASPERA_SCP_FILEPASS. For uploads (`--mode=send`), use `--file-crypt=encrypt`. For downloads (`--mode=recv`), use `--file-crypt=decrypt`.

Note: When a transfer to HSTS 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.

Server configuration

To configure the server to require client-side EAR and to require strong content protection passwords, run the following commands:

```
# asconfigurator -x "set_node_data;transfer_encryption_content_protection_required,true"
# asconfigurator -x "set_node_data;transfer_encryption_content_protection_strong_pass_required,true"
```

Note: These commands set the global configuration. Depending on your work flow, you might want to require client-side EAR and strong passwords for only specific users or groups.

5. For particularly sensitive content, do not store unecrypted content on any computer with network access.

   HSTS, HSTE, and Desktop Client include the **asprotect** and **asunprotect** command-line tools that can be used to encrypt and decrypt files. Use an external drive to physically move encrypted files between a network-connected computer and an unconnected computer on which the files can be unencrypted.

   - To encrypt a file before moving it to a computer with network access, run the following commands to set the encryption password and encrypt the file:

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

   - To download client-side-encrypted files without decrypting them immediately, run the transfer without decryption enabled (clear **Decrypt password-protected files downloaded** in the GUI or do not specify `--file-crypt=decrypt` on the **ascp** command line).
   - To decrypt encrypted files, run the following commands to set the encryption password and decrypt the file:

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

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 Aspera log file includes detailed transfer information and can be useful for review and support requests.

The log file is found in `/var/log/aspera.log`

If you find that logs are being overwritten before long transfers of many files are complete, you can increase the log size. For more information, see “Logs Overwritten Before Transfer Completes” on page 343.

Logging Client File System Activity on HSTS

HSTS can be configured to log operations on the server's file system that are performed from client applications (such as the HSTS in client mode, or Console).

The logging of specific file system operations is controlled with an <ascmd> element in `aspera.conf`, within which logging can be set to yes or no for each operation.

```
<ascmd>
    <log_cmd>
        <as_info>no</as_info>
        <as_ls>no</as_ls>
        <as_rm>no</as_rm>
        <as_du>no</as_du>
        <as_df>no</as_df>
        <as_mkdir>no</as_mkdir>
        <as_cp>no</as_cp>
        <as_mv>no</as_mv>
        <as_md5sum>no</as_md5sum>
    </log_cmd>
</ascmd>
```

As an example of **`asconfigurator`** usage, the following command specifies that any deletions from the server file system by user xeno are logged:

```
asconfigurator -x "set_user_data;user_name,xeno;ascmd_log_cmd_as_rm,yes"
```

The command generates this <ascmd> element in `aspera.conf`:

```
<ascmd>
    <log_cmd>
        <as_rm>yes</as_rm>
    </log_cmd>
</ascmd>
```

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.