Command Line Interface for IBM Aspera products

Laurent MARTIN

2023/11/13

Contents

Chapter 1

Introduction

Version : 4.15.0.pre

Laurent/2016-2023

This gem provides the `ascli` CLI (Command Line Interface) to IBM Aspera software.

`ascli` is a also great tool to learn Aspera APIs.

Ruby Gem: https://rubygems.org/gems/aspera-cli

Ruby Doc: https://www.rubydoc.info/gems/aspera-cli

Minimum required Ruby version: >= 2.6.

> Deprecation notice: the minimum Ruby version will be 3.0 in a future version.

Aspera APIs on IBM developer Link 2

Release notes: see CHANGELOG.md

A PDF version of this documentation is available here: docs/Manual.pdf.

1.1 BUGS, FEATURES, CONTRIBUTION

Refer to BUGS.md and CONTRIBUTING.md.

1.2 When to use and when not to use

`ascli` is designed to be used as a command line tool to:

- Execute commands remotely on Aspera products
- Transfer to/from Aspera products

So it is designed for:

- Interactive operations on a text terminal (typically, VT100 compatible), e.g. for maintenance
- Scripting, e.g. batch operations in (shell) scripts (e.g. cron job)

`ascli` can be seen as a command line tool integrating:

- A configuration file (config.yaml)
- Advanced command line options
- cURL (for REST calls)
- Aspera transfer (`ascp`)

If the need is to perform operations programmatically in languages such as: C, Go, Python, nodejs, ... then it is better to directly use Aspera APIs

- Product APIs (REST) : e.g. AoC, Faspex, node
- Transfer SDK : with gRPC interface and language stubs (C, C++, Python, .NET/C#, java, Ruby, etc...)

Using APIs (application REST API and transfer SDK) will prove to be easier to develop and maintain.

For scripting and ad'hoc command line operations, `ascli` is perfect.

1.3 Notations, Shell, Examples

Command line operations examples are shown using a shell such: `bash` or `zsh`.

Command line parameters in examples beginning with `my_`, like `my_param_value` are user-provided value and not fixed value commands.

`ascli` is an API Client toward the remote Aspera application Server (Faspex, HSTS, etc...)

Some commands will start an Aspera-based transfer (e.g. `upload`). The transfer is not directly implemented in `ascli`, rather `ascli` uses an external Aspera Client called Transfer Agents.

> Note: The transfer agent is a client for the remote Transfer Server (HSTS). The transfer agent may be local or remote... For example a remote Aspera Server may be used as a transfer agent (using node API). i.e. using option `--transfer=node`

Chapter 2

Quick Start

This section guides you from installation, first use and advanced use.

First, follow the section: Installation (Ruby, Gem, FASP) to start using `ascli`.

Once the gem is installed, `ascli` shall be accessible:

```
ascli --version
```

```
4.15.0.pre
```

2.1 First use

Once installation is completed, you can proceed to the first use with a demo server:

If you want to test with Aspera on Cloud, jump to section: Wizard

To test with Aspera demo transfer server, setup the environment and then test:

```
ascli config initdemo
```

```
ascli server browse /
```

```
+------------+--------+-----------+-------+---------------------------+-----------------------+
| zmode      | zuid   | zgid      | size  | mtime                     | name                  |
+------------+--------+-----------+-------+---------------------------+-----------------------+
| drwxr-xr-x | aspera | asperaweb | 90112 | 2023-04-05 15:31:21 +0200 | Upload                |
| dr-xr-xr-x | aspera | asperaweb | 4096  | 2022-10-27 16:08:16 +0200 | aspera-test-dir-large |
| dr-xr-xr-x | aspera | asperaweb | 4096  | 2022-10-27 16:08:17 +0200 | aspera-test-dir-small |
| dr-xr-xr-x | aspera | asperaweb | 4096  | 2022-10-27 16:08:17 +0200 | aspera-test-dir-tiny  |
+------------+--------+-----------+-------+---------------------------+-----------------------+
```

If you want to use `ascli` with another server, and in order to make further calls more convenient, it is advised to define a option preset for the server's authentication options. The following example will:

- create a option preset
- define it as default for `server` plugin
- list files in a folder
- download a file

```
ascli config preset update myserver --url=ssh://demo.asperasoft.com:33001 --username=asperaweb
↪  --password=my_password_here
```

```
updated: myserver
```

```
ascli config preset set default server myserver
```

```
updated: default &rarr; server to myserver
```

```
ascli server browse /aspera-test-dir-large
```

```
+------------+-----------+-----------+--------------+---------------------------+-------+
| zmode      | zuid      | zgid      | size         | mtime                     | name  |
+------------+-----------+-----------+--------------+---------------------------+-------+
| -r-xr-x--- | asperaweb | asperaweb | 104857600    | 2022-10-27 16:06:38 +0200 | 100MB |
| -r-xr-x--- | asperaweb | asperaweb | 10737418240  | 2022-10-27 16:08:12 +0200 | 10GB  |
| -r-xr-x--- | asperaweb | asperaweb | 500000000000 | 2022-10-27 16:06:26 +0200 | 500GB |
| -r-xr-x--- | asperaweb | asperaweb | 524288000    | 2022-10-27 14:53:00 +0200 | 500MB |
| -r-xr-x--- | asperaweb | asperaweb | 1048576000   | 2022-10-27 16:06:37 +0200 | 1GB   |
| -r-xr-x--- | asperaweb | asperaweb | 5368709120   | 2022-10-27 14:53:47 +0200 | 5GB   |
| -r-xr-x--- | asperaweb | asperaweb | 209715200    | 2022-10-27 14:52:56 +0200 | 200MB |
+------------+-----------+-----------+--------------+---------------------------+-------+
```

```
ascli server download /aspera-test-dir-large/200MB
```

```
Time: 00:00:02 ============================================ 100% 100 Mbps Time: 00:00:00
complete
```

2.2 Going further

Get familiar with configuration, options, commands : Command Line Interface.

Then, follow the section relative to the product you want to interact with (Aspera on Cloud, Faspex, ...) : Application Plugins

Chapter 3

Installation

It is possible to install either directly on the host operating system (Linux, macOS, Windows) or as a container (`docker`).

The direct installation is recommended and consists in installing:

- Ruby
- aspera-cli
- Aspera SDK (`ascp`)

Ruby version: >= 2.6.

> Deprecation notice: the minimum Ruby version will be 3.0 in a future version.

The following sections provide information on the various installation methods.

An internet connection is required for the installation. If you don't have internet for the installation, refer to section Installation without internet access.

3.1 Container

The container image is: martinlaurent/ascli. The container contains: Ruby, `ascli` and the Aspera Transfer SDK. To use the container, ensure that you have `podman` (or `docker`) installed.

```
podman --version
```

3.1.1 Container: quick start

Wanna start quickly ? With an interactive shell ? Execute this:

```
podman run --tty --interactive --entrypoint bash martinlaurent/ascli:latest
```

Then, execute individual `ascli` commands such as:

```
ascli conf init
ascli conf preset overview
ascli conf ascp info
ascli server ls /
```

That is simple, but there are limitations:

- Everything happens in the container
- Any generated file in the container will be lost on container (shell) exit. Including configuration files and downloaded files.
- No possibility to upload files located on the host system

3.1.2 Container: Details

The container image is built from this Dockerfile: the entry point is `ascli` and the default command is `help`.

If you want to run the image with a shell, execute with option: `--entrypoint bash`, and give argument `-l` (`bash` login option to override the `help` default argument)

The container can also be executed for individual commands like this: (add `ascli` commands and options at the end of the command line, e.g. `-v` to display the version)

```
podman run --rm --tty --interactive martinlaurent/ascli:latest
```

For more convenience, you may define a shell alias:

```
alias ascli='podman run --rm --tty --interactive martinlaurent/ascli:latest'
```

Then, you can execute the container like a local command:

```
ascli -v
```

```
4.15.0.pre
```

In order to keep persistency of configuration on the host, you should specify your user's config folder as a volume for the container. To enable write access, a possibility is to run as `root` in the container (and set the default configuration folder to `/home/cliuser/.aspera/ascli`). Add options:

```
--user root --env ASCLI_HOME=/home/cliuser/.aspera/ascli --volume
↪ $HOME/.aspera/ascli:/home/cliuser/.aspera/ascli
```

> Note: if you are using a `podman machine`, e.g. on macOS , make sure that the folder is also shared between the VM and the host, so that sharing is: container → VM → Host: `podman machine init ... --volume="/Users:/Users"`

As shown in the quick start, if you prefer to keep a running container with a shell and `ascli` available, you can change the entry point, add option:

```
--entrypoint bash
```

You may also probably want that files downloaded in the container are directed to the host. In this case you need also to specify the shared transfer folder as a volume:

```
--volume $HOME/xferdir:/xferfiles
```

> Note: ascli is run inside the container, so transfers are also executed inside the container and do not have access to host storage by default.

And if you want all the above, simply use all the options:

```
alias asclish="podman run --rm --tty --interactive --user root --env
↪ ASCLI_HOME=/home/cliuser/.aspera/ascli --volume
↪ $HOME/.aspera/ascli:/home/cliuser/.aspera/ascli --volume $HOME/xferdir:/xferfiles
↪ --entrypoint bash martinlaurent/ascli:latest"
```

```
export xferdir=$HOME/xferdir
mkdir -p $xferdir
chmod -R 777 $xferdir
mkdir -p $HOME/.aspera/ascli
asclish
```

3.1.3 Container: Sample start script

A convenience sample script is also provided: download the script `dascli` from the GIT repo :

> Note: If you have installed `ascli`, the script `dascli` can also be found: `cp $(ascli conf gem path)/../examples/dascli ascli`

Some environment variables can be set for this script to adapt its behavior:

env var	description	default	example
ASCLI_HOME	configuration folder (persistency)	`$HOME/.aspera/ascli`	`$HOME/.ascli_config`
docker_args	additional options to `podman`	<empty>	`--volume /Users:/Users`
image	container image name	martinlaurent/ascli	

env var	description	default	example
version	container image version	latest	4.8.0.pre

The wrapping script maps the folder $ASCLI_HOME on host to /home/cliuser/.aspera/ascli in the container. (value expected in the container). This allows having persistent configuration on the host.

To add local storage as a volume, you can use the env var docker_args:

Example of use:

```
curl -o ascli https://raw.githubusercontent.com/IBM/aspera-cli/main/examples/dascli
chmod a+x ascli
export xferdir=$HOME/xferdir
mkdir -p $xferdir
chmod -R 777 $xferdir
export docker_args="--volume $xferdir:/xferfiles"

./ascli conf init

echo 'Local file to transfer' > $xferdir/samplefile.txt
./ascli server upload /xferfiles/samplefile.txt --to-folder=/Upload
```

> Note: The local file (samplefile.txt) is specified relative to storage view from container (/xferfiles) mapped to the host folder $HOME/xferdir
>
> Note: Do not use too many volumes, as the AUFS limits the number.

3.1.4 Container: Offline installation

- First create the image archive:

```
podman pull martinlaurent/ascli
podman save martinlaurent/ascli|gzip>ascli_image_latest.tar.gz
```

- Then, on air-gapped system:

```
podman load -i ascli_image_latest.tar.gz
```

3.1.5 Container: **aspera.conf**

ascp's configuration file aspera.conf is located in the container at: /aspera_sdk/aspera.conf (see Dockerfile). As the container is immutable, it is not recommended to modify this file. If one wants to change the content, it is possible to tell ascp to use another file using ascp option -f, e.g. by locating it on the host folder $HOME/.aspera/ascli mapped to the container folder /home/cliuser/.aspera/ascli:

```
echo '<CONF/>' > $HOME/.aspera/ascli/aspera.conf
```

Then, tell ascp to use that other configuration file:

```
--transfer-info=@json:'{"ascp_args":["-f","/home/cliuser/.aspera/ascli/aspera.conf"]}'
```

3.1.6 Container: Singularity

Singularity is another type of use of container.

On Linux install:

```
dnf install singularity-ce
```

Build an image like this:

```
singularity build ascli.sif docker://martinlaurent/ascli
```

Then, start ascli like this:

```
singularity run ascli.sif
```

Or get a shell with access to `ascli` like this:

```
singularity shell ascli.sif
```

3.2 Ruby

Use this method to install on the native host.

A Ruby interpreter is required to run `ascli` or to use the gem and tool.

Required Ruby version: >= 2.6.

> Deprecation notice: the minimum Ruby version will be 3.0 in a future version.

Ruby can be installed using any method : rpm, yum, dnf, rvm, brew, Windows installer, ...

In priority, refer to the official Ruby documentation:

- Download Ruby
- Installation Guide

Else, refer to the following sections for a proposed method for specific operating systems.

The recommended installation method is `rvm` for Unix-like systems (Linux, AIX, macOS, Windows with cygwin, etc...). If the generic install is not suitable (e.g. Windows, no cygwin), you can use one of OS-specific install method. If you have a simpler better way to install Ruby : use it !

3.2.1 Generic: RVM: single user installation (not root)

Use this method which provides more flexibility.

Install `rvm`: follow https://rvm.io/ :

Execute the shell/curl command. As regular user, it install in the user's home: `~/.rvm`.

```
\curl -sSL https://get.rvm.io | bash -s stable
```

Follow on-screen instructions to install keys, and then re-execute the command.

If you keep the same terminal (not needed if re-login):

```
source ~/.rvm/scripts/rvm
```

It is advised to get one of the pre-compiled Ruby version, you can list with:

```
rvm list --remote
```

Install the chosen pre-compiled Ruby version:

```
rvm install 3.2.2
```

Ruby is now installed for the user, go to Gem installation.

3.2.2 Generic: RVM: global installation (as root)

Follow the same method as single user install, but execute as "root".

As root, it installs by default in /usr/local/rvm for all users and creates `/etc/profile.d/rvm.sh`. One can install in another location with :

```
curl -sSL https://get.rvm.io | bash -s -- --path /usr/local
```

As root, make sure this will not collide with other application using Ruby (e.g. Faspex). If so, one can rename the login script:

```
mv /etc/profile.d/rvm.sh /etc/profile.d/rvm.sh.ok
```

To activate Ruby (and ascli) later, source it:

```
source /etc/profile.d/rvm.sh.ok
```

```
rvm version
```

3.2.3 Windows: Installer

Manual installation:

- Navigate to https://rubyinstaller.org/ → Downloads.
- Download the latest Ruby installer with devkit. (Msys2 is needed to install some native extensions, such as `grpc`)
- Execute the installer which installs by default in: `C:\RubyVV-x64` (VV is the version number)
- At the end of the installation procedure, the Msys2 installer is automatically executed, select option 3 (msys and mingw)
- then install the aspera-cli gem and Aspera SDK (see next sections)

Automated installation, with internet access:

The ruby installer supports silent installation, to see the options, execute it with `/help`, or refer to the Ruby Installer FAQ

Download theruby installer executable from https://rubyinstaller.org/downloads/ and then install:

```
rubyinstaller-devkit-3.2.2-1-x64.exe /silent /currentuser /noicons /dir=C:\aspera-cli
```

Installation without network:

It is essentially the same procedure, but instead of retrieving files from internet, one copies the files from a machine with internet access, and then install from those archives:

- Download the `exe` ruby installer from https://rubyinstaller.org/downloads/
- Create an archive with necessary gems: https://help.rubygems.org/kb/rubygems/installing-gems-with-no-network

  ```
  gem install aspera-cli -N -i my_gems
  ```

  Zip the files `*.gem` from folder `repo/my_gems`
- Download the SDK from: https://ibm.biz/aspera_sdk

Create a Zip with all those files, and transfer to the target system.

Then, on the target system:

- Unzip the archive
- Execute the installer:

```
rubyinstaller-devkit-3.2.2-1-x64.exe /silent /currentuser /noicons /dir=C:\aspera-cli
```

- Install the gems:

```
gem install --force --local *.gem
```

- install the SDK

```
ascli conf ascp install --sdk-url=file:///sdk.zip
```

Note: An example of installation script is provided: docs/install.bat

3.2.4 macOS: pre-installed or **`brew`**

macOS 10.13+ (High Sierra) comes with a recent Ruby. So you can use it directly. You will need to install aspera-cli using `sudo` :

```
sudo gem install aspera-cli --pre
```

Alternatively, if you use Homebrew already you can install Ruby with it:

```
brew install ruby
```

3.2.5 Linux: package

If your Linux distribution provides a standard Ruby package, you can use it provided that the version supported.

Example: RHEL 8+, Rocky Linux 8+, Centos 8 Stream: with extensions to compile native gems

- Check available ruby versions:

```
dnf module list ruby
```

- If ruby was already installed with an older version, remove it:

```
dnf module -y reset ruby
```

- Install packages needed to build native gems:

```
dnf install -y make automake gcc gcc-c++ kernel-devel
```

- Enable the Ruby version you want:

```
dnf module -y enable ruby:3.1
dnf install -y ruby-devel
```

Other examples:

```
yum install -y ruby ruby-devel rubygems ruby-json
```

```
apt install -y ruby ruby-dev rubygems ruby-json
```

One can cleanup the whole yum-installed Ruby environment like this to uninstall:

```
gem uninstall -axI $(ls $(gem env gemdir)/gems/|sed -e 's/-[^-]*$//'|sort -u)
```

3.2.6 Other Unixes (AIX)

Ruby is sometimes made available as installable package through third party providers. For example for AIX, one can look at:

https://www.ibm.com/support/pages/aix-toolbox-open-source-software-downloads-alpha#R

If your Unix does not provide a pre-built Ruby, you can get it using one of those methods.

For instance to build from source, and install in `/opt/ruby` :

```
wget https://cache.ruby-lang.org/pub/ruby/2.7/ruby-2.7.2.tar.gz

gzip -d ruby-2.7.2.tar.gz

tar xvf ruby-2.7.2.tar

cd ruby-2.7.2

./configure --prefix=/opt/ruby

make ruby.imp

make

make install
```

If you already have a Java JVM on your system (`java`), it is possible to use `jruby`:

https://www.jruby.org/download

> Note: Using `jruby`, the startup time is longer than the native Ruby, but transfer speed is not impacted (executed by `ascp` binary).

3.3 **`aspera-cli`** gem

Once you have Ruby and rights to install gems: Install the gem and its dependencies:

```
gem install aspera-cli --pre
```

To upgrade to the latest version:

```
gem update aspera-cli
```

`ascli` checks every week if a new version is available and notify the user in a WARN log. To de-activate this feature, globally set the option `version_check_days` to `0`, or specify a different period in days.

To check if a new version is available (independently of `version_check_days`):

```
ascli conf check_update
```

3.4 FASP Protocol

Most file transfers will be done using the FASP protocol, using `ascp`. Only two additional files are required to perform an Aspera Transfer, which are part of Aspera SDK:

- `ascp`
- aspera-license (in same folder, or ../etc)

This can be installed either be installing an Aspera transfer software, or using an embedded command:

```
ascli conf ascp install
```

If a local SDK installation is preferred instead of fetching from internet: one can specify the location of the SDK file:

```
curl -Lso sdk.zip https://ibm.biz/aspera_sdk
```

```
ascli conf ascp install --sdk-url=file:///sdk.zip
```

The format is: `file:///<path>`, where `<path>` can be either a relative path (not starting with `/`), or an absolute path.

If the embedded method is not used, the following packages are also suitable:

- IBM Aspera Connect Client (Free)
- IBM Aspera Desktop Client (Free)
- IBM Aspera High Speed Transfer Server (Licensed)
- IBM Aspera High Speed Transfer EndPoint (Licensed)

For instance, Aspera Connect Client can be installed by visiting the page: https://www.ibm.com/aspera/connect/.

`ascli` will detect most of Aspera transfer products in standard locations and use the first one found. Refer to section FASP for details on how to select a client or set path to the FASP protocol.

Several methods are provided to start a transfer. Use of a local client (`direct` transfer agent) is one of them, but other methods are available. Refer to section: Transfer Agents

3.5 Installation in air gapped environment

> Note: no pre-packaged version is provided.

A method to build one is provided here:

The procedure:

- Follow the non-root installation procedure with RVM, including gem
- Archive (zip, tar) the main RVM folder (includes ascli):

```
cd $HOME && tar zcvf rvm-ascli.tgz .rvm
```

- Get the Aspera SDK.

```
ascli conf --show-config --fields=sdk_url
```

- Download the SDK archive from that URL.

```
curl -Lso sdk.zip https://ibm.biz/aspera_sdk
```

- Transfer those 2 files to the target system
- On target system

```
cd $HOME

tar zxvf rvm-ascli.tgz

source ~/.rvm/scripts/rvm

ascli conf ascp install --sdk-url=file:///sdk.zip
```

- Add those lines to shell init (`.profile`)

```
source ~/.rvm/scripts/rvm
```

Chapter 4

Command Line Interface: **ascli**

The `aspera-cli` Gem provides a command line interface (CLI) which interacts with Aspera Products (mostly using REST APIs):

- IBM Aspera High Speed Transfer Server (FASP and Node)
- IBM Aspera on Cloud (including ATS)
- IBM Aspera Faspex
- IBM Aspera Shares
- IBM Aspera Console
- IBM Aspera Orchestrator
- and more...

`ascli` provides the following features:

- Supports most Aspera server products (on-premise and SaaS)
- Any command line options (products URL, credentials or any option) can be provided on command line, in configuration file, in env var, in files
- Supports Commands, Option values and Parameters shortcuts
- FASP Transfer Agents can be: local `ascp`, or Connect Client, or any transfer node
- Transfer parameters can be altered by modification of transfer-spec, this includes requiring multi-session
- Allows transfers from products to products, essentially at node level (using the node transfer agent)
- Supports FaspStream creation (using Node API)
- Supports Watchfolder creation (using Node API)
- Additional command plugins can be written by the user
- Supports download of faspex and Aspera on Cloud ”external” links
- Supports ”legacy” ssh based FASP transfers and remote commands (ascmd)

Basic usage is displayed by executing:

```
ascli -h
```

Refer to sections: Usage.

Not all `ascli` features are fully documented here, the user may explore commands on the command line.

4.1 **ascp** command line

If you want to use `ascp` directly as a command line, refer to IBM Aspera documentation of either Desktop Client, Endpoint or Transfer Server where a section on `ascp` can be found.

Using `ascli` with plugin `server` for command line gives advantages over `ascp`:

- automatic resume on error
- configuration file
- choice of transfer agents
- integrated support of multi-session

Moreover all `ascp` options are supported either through transfer spec parameters and with the possibility to provide `ascp` arguments directly when the `direct` agent is used (`ascp_args`).

4.2 Command line parsing, Special Characters

`ascli` is typically executed in a shell, either interactively or in a script. `ascli` receives its arguments from this shell (through Operating System).

4.2.1 Shell parsing for Unix-like systems: Linux, macOS, AIX

Linux command line parsing is easy: It is fully documented in the shell's documentation.

On Unix-like environments, this is typically a POSIX shell (bash, zsh, ksh, sh). In this environment the shell parses the command line, possibly replacing variables, etc... See bash shell operation. The shell builds the list of arguments and then `fork`/`exec` Ruby with that list. Ruby receives a list parameters from shell and gives it to `ascli`. So special character handling (quotes, spaces, env vars, ...) is handled by the shell for any command executed.

4.2.2 Shell parsing for Windows

MS Windows command line parsing is not handled by the shell (`cmd.exe`), not handled by the operating system, but it is handled by the executable. Typically, Windows executables use the microsoft library for this parsing.

As far as `ascli` is concerned: the executable is Ruby. It has its own parsing algorithm, close to a Linux shell parsing.

Thanksfully, `ascli` provides a command to check the value of an argument after parsing: `config echo`. One can also run `ascli` with option `--log-level=debug` to display the command line after parsing.

The following examples give the same result on Windows:

- single quote protects the double quote

```
conf echo @json:'{"url":"https://..."}'
```

- triple double quotes are replaced with a single double quote

```
conf echo @json:{"""url""":"""https://..."""}
```

- double quote is escaped with backslash within double quotes

```
conf echo @json:"{\"url\":\"https://...\"}"
```

More details: on Windows, `cmd.exe` is typically used to start . `cmd.exe` handles some special characters: ^"<>|%&. Basically it handles I/O redirections (<>|), shell variables (%), multiple commands (&) and handles those special characters from the command line. Eventually, all those special characters are removed from the command line unless escaped with ^ or ". " are kept and given to the program.

Then, Windows `CreateProcess` is called with just the whole command line as a single string, unlike Unix-like systems where the command line is split into arguments by the shell.

It's up to the program to split arguments:

- Windows: How Command Line Parameters Are Parsed
- Understand Quoting and Escaping of Windows Command Line Arguments

is a Ruby program, so Ruby parses the command line into arguments and provides them to the program. Ruby vaguely follows the Microsoft C/C++ parameter parsing rules. (See `w32_cmdvector` in Ruby source `win32.c`) :

- space characters: split arguments (space, tab, newline)
- backslash: \ escape single special character
- globbing characters: `*?[]{}` for file globbing
- double quotes: "
- single quotes: '

4.2.3 Extended Values (JSON, Ruby, ...)

Some of the `ascli` parameters are expected to be Extended Values, i.e. not a simple `String`, but a complex structure (`Hash`, `Array`). Typically, the `@json:` modifier is used, it expects a JSON string. JSON itself has some special syntax: for example `"` is used to enclose a `String`.

4.2.4 Testing Extended Values

In case of doubt of argument values after parsing, one can test using command `config echo`. `config echo` takes exactly one argument which can use the Extended Value syntax. Unprocessed command line arguments are shown in the error message.

Example: The shell parses three arguments (as `String`: 1, 2 and 3), so the additional two arguments are not processed by the `echo` command.

```
ascli conf echo 1 2 3
```

```
"1"
ERROR: Argument: unprocessed values: ["2", "3"]
```

`config echo` displays the value of the first argument using Ruby syntax: it surrounds a string with `"` and add `\` before special characters.

> Note: It gets its value after shell command line parsing and `ascli` extended value parsing.

In the following examples (using a POSIX shell, such as `bash`), several sample commands are provided when equivalent. For all example, most of special character handling is not specific to `ascli`: It depends on the underlying syntax: shell , JSON, etc... Depending on the case, a different `format` is used to display the actual value.

For example, in the simple string `Hello World`, the space character is special for the shell, so it must be escaped so that a single value is represented.

Double quotes are processed by the shell to create a single string argument. For POSIX shells, single quotes can also be used in this case, or protect the special character (space) with a backslash.

```
ascli conf echo "Hello World" --format=text
ascli conf echo 'Hello World' --format=text
ascli conf echo Hello\ World --format=text
```

```
Hello World
```

4.2.5 Using a shell variable, parsed by shell, in an extended value

To be evaluated by shell, the shell variable must not be in single quotes. Even if the variable contains spaces it makes only one argument to `ascli` because word parsing is made before variable expansion by shell.

> Note: we use a shell variable here: the variable is not necessarily an environment variable (`export`).

```
MYVAR="Hello World"
ascli conf echo @json:'{"title":"'$MYVAR'"}' --format=json
ascli conf echo @json:{\"title\":\"$MYVAR\"} --format=json
```

```
{"title":"Hello World"}
```

4.2.6 Double quote in strings in command line

Double quote is a shell special character. Like any shell special character, it can be protected either by preceding with a backslash or by enclosing in a single quote.

```
ascli conf echo \"
ascli conf echo '"'
```

```
"
```

Double quote in JSON is a little tricky because `"` is special both for the shell and JSON. Both shell and JSON syntax allow to protect `"`, but only the shell allows protection using single quote.

```
ascli conf echo @json:'"\""' --format=text
ascli conf echo @json:\"\\\"\" --format=text
ascli conf echo @ruby:\'\"\' --format=text
```

```
"
```

Here a single quote or a backslash protects the double quote to avoid shell processing, and then an additional \ is added to protect the " for JSON. But as \ is also shell special, then it is protected by another \.

4.2.7 Shell and JSON or Ruby special characters in extended value

Construction of values with special characters is done like this:

- First select a syntax to represent the extended value, e.g. JSON or Ruby
- Write the expression using this syntax, for example, using JSON:

```
{"title":"Test \" ' & \\"}
```

or using Ruby:

```
{"title"=>"Test \" ' & \\"}
{'title'=>%q{Test " ' & \\}}
```

Both " and \ are special characters for JSON and Ruby and can be protected with \ (unless Ruby's extended single quote notation %q is used).

- Then, since the value will be evaluated by shell, any shell special characters must be protected, either using preceding \ for each character to protect, or by enclosing in single quote:

```
ascli conf echo @json:{\"title\":\"Test\ \\\"\ \'\ \&\ \\\\\"} --format=json
ascli conf echo @json:'{"title":"Test \" '\'' & \\"}' --format=json
ascli conf echo @ruby:"{'title'=>%q{Test \" ' & \\\\}}" --format=json
```

```
{"title":"Test \" ' & \\"}
```

4.2.8 Reading special characters interactively

If `ascli` is used interactively (a user typing on terminal), it is easy to require the user to type values:

```
ascli conf echo @ruby:"{'title'=>gets.chomp}" --format=json
```

`gets` is Ruby's method of terminal input (terminated by `\n`), and `chomp` removes the trailing `\n`.

4.2.9 Command line arguments from a file

If you need to provide a list of command line argument from lines that are in a file, on Linux you can use the `xargs` command:

```
xargs -a lines.txt -d \\n ascli conf echo
```

This is equivalent to execution of:

```
ascli conf echo [line1] [line2] [line3] ...
```

If there are spaces in the lines, those are not taken as separator, as we provide option `-d \\n` to `xargs`.

4.2.10 Extended value using special characters read from environmental variables or files

Using a text editor or shell: create a file `title.txt` (and env var) that contains exactly the text required: `Test " ' & \`:

```
export MYTITLE='Test " '\'' & \'
echo -n $MYTITLE > title.txt
```

Using those values will not require any escaping of characters since values do not go through shell or JSON parsing.

If the value is to be assigned directly to an option of ascli, then you can directly use the content of the file or env var using the `@file:` or `@env:` readers:

```
ascli conf echo @file:title.txt --format=text
ascli conf echo @env:MYTITLE --format=text
```

```
Test " ' & \
```

If the value to be used is in a more complex structure, then the `@ruby:` modifier can be used: it allows any Ruby code in expression, including reading from file or env var. In those cases, there is no character to protect because values are not parsed by the shell, or JSON or even Ruby.

```
ascli conf echo @ruby:"{'title'=>File.read('title.txt')}" --format=json
ascli conf echo @ruby:"{'title'=>ENV['MYTITLE']}" --format=json
```

```
{"title":"Test \" ' & \\"}
```

4.3 Commands, Options, Positional Arguments

Command line arguments are the units of command line typically separated by spaces (the `argv` of C). The tokenization of the command line is typically done by the shell, refer to the previous section Command Line Parsing.

`ascli` considers three types of command line arguments:

- Commands
- Options
- Positional Arguments

For example:

```
ascli command subcommand --option-name=VAL1 VAL2
```

- executes command: `command subcommand`
- with one option: `option_name` and its value: `VAL1`
- the command has one additional positional argument: `VAL2`

If the value of a command, option or argument is constrained by a fixed list of values, then it is possible to use a few of the first letters of the value, provided that it uniquely identifies the value. For example `ascli conf pre ov` is the same as `ascli config preset overview`.

The value of options and arguments is evaluated with the Extended Value Syntax.

4.3.1 Commands

Commands are typically entity types or verbs to act on those entities.

Example:

```
ascli conf ascp info
```

- `ascli` is the executable executed by the shell
- `conf` is the first level command, and is also the name f the plugin to be used
- `ascp` is the second level command, and is also the name of the component (singleton)
- `info` is the third level command, and is also the action to be performed

Typically, commands are located at the beginning of the command line. Order is significant. The provided command must match one of the supported commands in the given context. If wrong, or no command is provided when expected, an error message is displayed and the list of supported commands is displayed.

Some sub-commands appear after entity selection, e.g. `ascli aoc admin res node do 8669 browse /`: `browse` is a sub-command of `node`.

4.3.2 Options

All options, e.g. `--log-level=debug`, are command line arguments that:

- start with `--`
- have a name, in lowercase, using `-` as word separator in name (e.g. `--log-level=debug`)
- have a value, separated from name with a `=`

- can be used by prefix, provided that it is unique. E.g. `--log-l=debug` is the same as `--log-level=debug` (avoid)

Exceptions:

- some options accept a short form, e.g. `-Ptoto` is equivalent to `--preset=toto`, refer to the manual or `-h`.
- some options (flags) don't take a value, e.g. `-r`
- the special option `--` stops option processing and is ignored, following command line arguments are taken as arguments, including the ones starting with a `-`. Example:

```
ascli config echo -- --sample
```

```
"--sample"
```

Note: Here, `--sample` is taken as an argument, and not as an option, due to `--`.

Options may have an (hardcoded) default value.

Options can be placed anywhere on command line and evaluated in order.

Options are typically:

- optional : to change the default behavior
- mandatory : so they can be placed in a config file, for example: connection information

The value for any options can come from the following locations (in this order, last value evaluated overrides previous value):

- Configuration file
- Environment variable
- Command line

Environment variable starting with prefix: ASCLI_ are taken as option values, e.g. `ASCLI_OPTION_NAME` is for `--option-name`.

Options values can be displayed for a given command by providing the `--show-config` option: `ascli node --show-config`

Parameters are typically designed as options if:

- it is a mandatory parameters that would benefit from being set in a config file or environment variable
- it is optional

4.3.3 Positional Arguments

Positional Arguments are typically mandatory values for a command, such as entity creation data.

It could also be designed as an option, but since it is mandatory and typically creation parameters need not be set in a configuration file, then it is better to use a positional argument, and not define specific options.

The advantages of using a positional argument instead of an option for the same are that the command line is shorter(no option name, just the position) and the value is clearly mandatory.

The disadvantage is that it is not possible to define a default value in a config file or environment variable like for options. Nevertheless, Extended Values syntax is supported, so it is possible to retrieve a value from the config file or environment variable.

If a Positional Arguments begins with `-`, then either use the `@val:` syntax (see Extended Values), or use the `--` separator (see above).

Very few positional arguments are optional, they are located at the end of the command line.

4.4 Interactive Input

Some options and parameters are mandatory and other optional. By default, `ascli` will ask for missing mandatory options or parameters for interactive execution.

The behavior can be controlled with:

- `--interactive=<yes|no>` (default=yes if STDIN is a terminal, else no)
  - yes : missing mandatory parameters/options are asked to the user
  - no : missing mandatory parameters/options raise an error message
- `--ask-options=<yes|no>` (default=no)
  - optional parameters/options are asked to user

4.5 Output

Command execution will result in output (terminal, stdout/stderr). The information displayed depends on the action.

4.5.1 Types of output data

Depending on action, the output will contain:

- `single_object` : displayed as a 2 dimensional table: one line per attribute, first column is attribute name, and second is attribute value. Nested hashes are collapsed.
- `object_list` : displayed as a 2 dimensional table: one line per item, one column per attribute.
- `value_list` : a table with one column.
- `empty` : nothing
- `status` : a message
- `other_struct` : a complex structure that cannot be displayed as an array

4.5.2 Format of output

By default, result of type single_object and object_list are displayed using format `table`. The table style can be customized with parameter: `table_style` (horizontal, vertical and intersection characters) and is `:.:` by default.

In a table format, when displaying "objects" (single, or list), by default, sub object are flattened (option `flat_hash`). So, object `{"user":{"id":1,"name":"toto"}}` will have attributes: `user.id` and `user.name`. Setting `flat_hash` to `false` will only display one field: `user` and value is the sub `Hash`. When in flatten mode, it is possible to filter fields by "dotted" field name.

Object lists are displayed one per line, with attributes as columns. Single objects are transposed: one attribute per line. If transposition of single object is not desired, use option: `transpose_single` set to `no`.

The style of output can be set using the `format` parameter, supporting:

- `text` : Value as String
- `table` : Text table
- `ruby` : Ruby code
- `json` : JSON code
- `jsonpp` : JSON pretty printed
- `yaml` : YAML
- `csv` : Comma Separated Values

4.5.3 Entity identifier

When a command is executed on a single entity, the entity is identified by a unique identifier that follows the command: e.g. `ascli aoc admin res user show 1234` where 1234 is the user's identifier.

> Note: The legacy option `id` is deprecated: `--id=1234` as it does not provide the possibility to have sub-entities.

Only some commands provide the following capability: If the entity can also be uniquely identified by a name, then the name can be used instead of the identifier, using the percent selector: `ascli aoc admin res user show %name:john` where `john` is the user name.

Syntax: `%<field>:<value>`

4.5.4 Verbosity of output

Output messages are categorized in 3 types:

- `info` output contain additional information, such as number of elements in a table
- `data` output contain the actual output of the command (object, or list of objects)
- `erroroutput` contain error messages

The option `display` controls the level of output:

- `info` displays all messages: `info`, `data`, and `error`
- `data` display `data` and `error` messages
- `error` display only error messages.

By default, secrets are removed from output: option `show_secrets` defaults to `no`, unless `display` is `data`, to allows piping results. To hide secrets from output, set option `show_secrets` to `no`.

4.5.5 Option: **`fields`**: Selection of output object properties

By default, a table output will display one line per entry, and columns for each properties. Depending on the command, columns may include by default all properties, or only some selected properties. It is possible to define specific columns to be displayed, by setting the `fields` option.

The `fields` option can be either a comma separated list, or an extended value array.

Elements of the list can be:

- `DEF` : default display of columns (that's the default, when not set)
- `ALL` : all columns available
- -property : remove property from the current list
- property : add property to the current list
- a ruby `RegEx` : using `@ruby:'/.../'`

Examples:

- `a,b,c` : the list of attributes specified as a comma separated list
- `@json:'["a","b","c"]'` : Array extended value: same as above
- `DEF,-a,b` : default property list, remove `a` and add `b`
- `@ruby:'/^server/'` : Display all properties whose name begin with `server`

4.5.6 Option: **`select`**: Filter on columns values for **`object_list`**

Table output can be filtered using option `select`. This parameter is either a `Hash` or `Proc`. The `Proc` takes as argument a line in the table.

Example:

```
ascli aoc admin res user list --fields=name,email,ats_admin --query=@json:'{"sort":"name"}'
 ↪  --select=@json:'{"ats_admin":true}'
```

```
+------------------------------+----------------------------------+-----------+
|             name             |              email               | ats_admin |
+------------------------------+----------------------------------+-----------+
| John Curtis                  | john@example.com                 | true      |
| Laurent Martin               | laurent@example.com              | true      |
+------------------------------+----------------------------------+-----------+
```

> Note: `select` filters elements from the result of command, while the `query` parameters gives filtering parameters to the API when listing elements.

In above example, the same result is obtained with option:

```
--select=@ruby:'->(u){u["ats_admin"]}'
```

4.6 Extended Value Syntax

Most options and arguments are specified by a simple string (e.g. username or url). Sometime it is convenient to read a value from a file: for example read the PEM value of a private key, or a list of files. Some options expect a more complex value such as `Hash` or `Array`.

The Extended Value Syntax allows to specify such values and even read values from other sources than the command line itself.

The syntax is:

```
<0 or more decoders><some text value or nothing>
```

Decoders act like a function with its parameter on right hand side and are recognized by the prefix: `@` and suffix `:`

The following decoders are supported:

decoder	parameter	returns	description
base64	String	String	decode a base64 encoded string
csvt	String	Array	decode a titled CSV value
env	String	String	read from a named env var name, e.g.--password=@env:MYPASSVAR
file	String	String	read value from specified file (prefix ~/ is replaced with the users home folder), e.g. `--key=@file:~/.ssh/mykey`
json	String	any	decode JSON values (convenient to provide complex structures)
lines	String	Array	split a string in multiple lines and return an array
list	String	Array	split a string in multiple items taking first character as separator and return an array
none	None	Nil	A null value
path	String	String	performs path expansion on specified path (prefix ~/ is replaced with the users home folder), e.g. `--config-file=@path:~/sample_config.yml`
preset	String	Hash	get whole option preset value by name. Sub-values can also be used using `.` as separator. e.g. `foo.bar` is `conf[foo][bar]`
extend	String	String	evaluates embedded extended value syntax in string
ruby	String	any	execute specified Ruby code
secret	None	String	Ask password interactively (hides input)
stdin	None	String	read from stdin (no value on right)
uri	String	String	read value from specified URL, e.g. `--fpac=@uri:http://serv/f.pac`
val	String	String	prevent decoders on the right to be decoded. e.g. `--key=@val:@file:foo` sets the option `key` to value `@file:foo`.
zlib	String	String	un-compress data

To display the result of an extended value, use the `config echo` command.

The `extend` decoder is useful to evaluate embedded extended value syntax in a string. It expects a `@` to close the embedded extended value syntax.

Example: Create a `Hash` value with the convenient `@json:` decoder:

```
ascli config echo @json:'{"key1":"value1","key2":"value2"}'
```

Example: read the content of the specified file, then, base64 decode, then unzip:

```
ascli config echo @zlib:@base64:@file:myfile.dat
```

Example: Create a `Hash` value with one key and the value is read from a file:

```
ascli config echo @ruby:'{"token_verification_key"=>File.read("mykey.txt")}'
```

Example: read a csv file and create an `Array` of `Hash` for bulk provisioning:

```
cat test.csv
```

```
name,email
lolo,laurent@example.com
toto,titi@tutu.tata
```

```
ascli config echo @csvt:@file:test.csv
```

```
+------+--------------------+
| name |        email       |
+------+--------------------+
| lolo | laurent@example.com |
| toto | titi@tutu.tata      |
+------+--------------------+
```

Example: create a JSON with values coming from a preset named ”config” of config file

```
ascli config echo @json:@extend:'{"hello":true,"version":"@preset:config.version@"}'
```

```
+---------+-----------+
| key     | value     |
+---------+-----------+
| hello   | true      |
| version | 4.14.0    |
+---------+-----------+
```

4.7 Configuration and Persistency Folder

`ascli` configuration and other runtime files (token cache, file lists, persistency files, SDK) are stored by default in `[User's home folder]/.aspera/ascli`.

> Note: `[User's home folder]` is found using Ruby's `Dir.home` (`rb_w32_home_dir`). It uses the `HOME` env var primarily, and on MS Windows it also looks at `%HOMEDRIVE%%HOMEPATH%` and `%USERPROFILE%`. `ascli` sets the env var `%HOME%` to the value of `%USERPROFILE%` if set and exists. So, on Windows `%USERPROFILE%` is used as it is more reliable than `%HOMEDRIVE%%HOMEPATH%`.

The configuration folder can be displayed using :

```
ascli config folder
```

```
/Users/kenji/.aspera/ascli
```

It can be overridden using option `home`.

Example (Windows):

```
set ASCLI_HOME=C:\Users\Kenji\.aspera\ascli

ascli config folder

C:\Users\Kenji\.aspera\ascli
```

When OAuth is used (AoC, Faspex4 api v4, Faspex5) `ascli` keeps a cache of generated bearer tokens in folder `persist_store` in configuration folder by default. Option `cache_tokens` (yes/no) allows to control if Oauth tokens are cached on file system, or generated for each request. The command `config flush_tokens` clears that cache. Tokens are kept on disk for a maximum of 30 minutes (`TOKEN_CACHE_EXPIRY_SEC`) and garbage collected after that. When a token has expired, then a new token is generated, either using a refresh_token if it is available, or by the default method.

4.8 Temporary files

Some temporary files may be needed during runtime. The temporary folder may be specified with option: `temp_folder`. Temporary files are deleted at the end of execution unless option: `clean_temp` is set to `no`.

4.9 Configuration file

On the first execution of `ascli`, an empty configuration file is created in the configuration folder. Nevertheless, there is no mandatory information required in this file, the use of it is optional as any option can be provided on the command line.

Although the file is a standard YAML file, `ascli` provides commands to read and modify it using the `config` command.

All options for `ascli` can be set on command line, or by env vars, or using option presets in the configuration file.

A configuration file provides a way to define default values, especially for authentication parameters, thus avoiding to always having to specify those parameters on the command line.

The default configuration file is: `$HOME/.aspera/ascli/config.yaml` (this can be overridden with option `--config-file=path` or equivalent env var).

The configuration file is simply a catalog of pre-defined lists of options, called: option presets. Then, instead of specifying some common options on the command line (e.g. address, credentials), it is possible to invoke the ones of a option preset (e.g. `mypreset`) using the option: `-Pmypreset` or `--preset=mypreset`.

4.9.1 Option preset

A option preset is simply a collection of parameters and their associated values in a named section in the configuration file.

A named option preset can be modified directly using `ascli`, which will update the configuration file :

```
ascli config preset set|delete|show|initialize|update <option preset>
```

The command `update` allows the easy creation of option preset by simply providing the options in their command line format, e.g. :

```
ascli config preset update demo_server --url=ssh://demo.asperasoft.com:33001 --username=asperaweb
↪  --password=my_password_here --ts=@json:'{"precalculate_job_size":true}'
```

- This creates a option preset `demo_server` with all provided options.

The command `set` allows setting individual options in a option preset.

```
ascli config preset set demo_server password my_password_here
```

The command `initialize`, like `update` allows to set several parameters at once, but it deletes an existing configuration instead of updating it, and expects a `Hash` Extended Value.

```
ascli config preset initialize demo_server @json:'{"url":"ssh://demo.asperasoft.com:33001",
↪  "username":"asperaweb","password":"my_pass_here","ts":{"precalculate_job_size":true}}'
```

A full terminal based overview of the configuration can be displayed using:

```
ascli config preset over
```

A list of option preset can be displayed using:

```
ascli config preset list
```

A good practice is to not manually edit the configuration file and use modification commands instead. If necessary, the configuration file can opened in a text editor with:

```
ascli config open
```

> Note: this starts the editor specified by env var `EDITOR` if defined.

Older format for commands are still supported:

```
ascli config preset set|delete|show|initialize|update <name>
ascli config preset over
ascli config preset list
```

4.9.2 Special Option preset: config

This preset name is reserved and contains a single key: `version`. This is the version of `ascli` which created the file.

4.9.3 Special Option preset: default

This preset name is reserved and contains an array of key-value , where the key is the name of a plugin, and the value is the name of another preset.

When a plugin is invoked, the preset associated with the name of the plugin is loaded, unless the option --no-default (or -N) is used.

> Note: Special plugin name: `config` can be associated with a preset that is loaded initially, typically used for default values.

Operations on this preset are done using regular `config` operations:

```
ascli config preset set default _plugin_name_ _default_preset_for_plugin_
```

```
ascli config preset get default _plugin_name_
```

```
"_default_preset_for_plugin_"
```

4.9.4 Plugin: **config**: Configuration

Plugin `config` provides general commands for `ascli`:

- Option preset, config file operations
- wizard
- vault
- ascp

The default preset for `config` is read for any plugin invocation, this allows setting global options, such as `--log-level` or `--interactive`. When `ascli` starts, it looks for the `default` Option preset and checks the value for `config`. If set, it loads the options independently of the plugin used.

> Note: If no global default is set by the user, `ascli` will use `global_common_defaults` when setting global parameters (e.g. `conf ascp use`)

> Note: If you don't know the name of the global preset, you can use `GLOBAL` to refer to it.

Show current default (global) Option preset (`config` plugin):

```
$ ascli conf preset get default config
global_common_defaults
```

```
ascli conf preset set GLOBAL version_check_days 0
```

If the default global Option preset is not set, and you want to use a different name:

```
ascli conf preset set my_common_defaults version_check_days 0
ascli conf preset set default config my_common_defaults
```

4.9.5 Config sample commands

```
config ascp connect info 'Aspera Connect for Windows'
config ascp connect list
config ascp connect version 'Aspera Connect for Windows' download 'Windows Installer'
 ↪  --to-folder=.
config ascp connect version 'Aspera Connect for Windows' list
config ascp connect version 'Aspera Connect for Windows' open documentation
config ascp errors
config ascp info --sdk-folder=Tsdk_test_dir
config ascp install
config ascp install --sdk-folder=Tsdk_test_dir
config ascp products list
config ascp products use 'Aspera Connect'
config ascp show
config ascp spec
config ascp use /usr/bin/ascp
config check_update
config coffee
config coffee --ui=text
config detect https://faspex4.example.com/path
config detect https://faspex5.example.com/path
config detect https://node.example.com/path
config detect https://shares.example.com/path shares
config detect my_org aoc
config doc
config doc transfer-parameters
config echo @base64:SGVsbG8gV29ybGQK
config echo @csvt:@stdin:
config echo @env:USER
config echo @lines:@stdin:
config echo @list:,1,2,3
config echo @secret:
config echo @uri:/etc/hosts
config echo @uri:file:/etc/hosts
config echo @uri:http://ifconfig.me
config echo @uri:https://ifconfig.me
config echo @vault:mypreset.password
config echo @zlib:@stdin:
config echo hello
config email_test --notif-to=my_email_external
config flush_tokens
config folder
config gem name
config gem path
config gem version
config genkey mykey
config genkey mykey 4096
config initdemo
config open
config plugin create mycommand T
config plugin list
config preset delete conf_name
config preset initialize conf_name @json:'{"p1":"v1","p2":"v2"}'
config preset list
config preset overview
config preset set conf_name param value
config preset set default shares conf_name
config preset show conf_name
config preset unset conf_name param
config preset update conf_name --p1=v1 --p2=v2
config proxy_check --fpac=@file:examples/proxy.pac https://eudemo.asperademo.com
 ↪  --proxy-credentials=@list:,user,pass
config vault create mylabel @json:'{"password":"my_password_here","description":"my secret"}'
```

```
config vault delete mylabel
config vault list
config vault show mylabel
config wizard https://console.example.com/path console
config wizard https://faspex4.example.com/path faspex --username=test --password=test
config wizard https://faspex5.example.com/path faspex5 --pkeypath=my_private_key
config wizard https://node.example.com/path node --username=test --password=test
config wizard https://orch.example.com/path orchestrator --username=test --password=test
config wizard https://shares.example.com/path shares --username=test --password=test
config wizard my_org aoc --pkeypath= --username=my_user_email
config wizard my_org aoc --pkeypath= --username=my_user_email --use-generic-client=yes
```

4.9.6 Format of file

The configuration file is a `Hash` in a YAML file. Example:

```
config:
  version: 0.3.7
default:
  config: cli_default
  server: demo_server
cli_default:
  interactive: no
demo_server:
  url: ssh://demo.asperasoft.com:33001
  username: asperaweb
  password: my_password_here
```

We can see here:

- The configuration was created with `ascli` version 0.3.7
- the default option preset to load for `server` plugin is : `demo_server`
- the option preset `demo_server` defines some parameters: the URL and credentials
- the default option preset to load in any case is : `cli_default`

Two option presets are reserved:

- `config` contains a single value: `version` showing the version used to create the configuration file. It is used to check compatibility.
- `default` is reserved to define the default option preset name used for known plugins.

The user may create as many option presets as needed. For instance, a particular option preset can be created for a particular application instance and contain URL and credentials.

Values in the configuration also follow the Extended Value Syntax.

> Note: if the user wants to use the Extended Value Syntax inside the configuration file, using the `config preset update` command, the user shall use the `@val:` prefix. Example:

```
ascli config preset set my_aoc_org private_key @val:@file:"$HOME/.aspera/ascli/my_private_key"
```

This creates the option preset:

```
...
my_aoc_org:
  private_key: @file:"/Users/laurent/.aspera/ascli/my_private_key"
...
```

So, the key file will be read only at execution time, but not be embedded in the configuration file.

4.9.7 Evaluation order of options

Some options are global, some options are available only for some plugins. (the plugin is the first level command).

Options are loaded using this algorithm:

- If option `--no-default` (or `-N`) is specified, then no default value is loaded for the plugin

- else it looks for the name of the plugin as key in section `default`, the value is the name of the default option preset for it, and loads it.
- If option `--preset=<name or extended value hash>` is specified (or `-Pxxxx`), this reads the option preset specified from the configuration file, or if the value is a `Hash`, it uses it as options values.
- Environment variables are evaluated
- Command line options are evaluated

Parameters are evaluated in the order of command line.

To avoid loading the default option preset for a plugin, use: `-N`

On command line, words in parameter names are separated by a dash, in configuration file, separator is an underscore. E.g. --xxx-yyy on command line gives xxx_yyy in configuration file.

The main plugin name is `config`, so it is possible to define a default option preset for the main plugin with:

```
ascli config preset set cli_default interactive no
```

```
ascli config preset set default config cli_default
```

A option preset value can be removed with `unset`:

```
ascli config preset unset cli_default interactive
```

Example: Define options using command line:

```
ascli -N --url=_url_here_ --password=my_password_here --username=_name_here_ node --show-config
```

Example: Define options using a `Hash`:

```
ascli -N
↪  --preset=@json:'{"url":"_url_here_","password":"my_password_here","username":"_name_here_"}'
↪  node --show-config
```

4.9.8 Wizard

The wizard is a command that asks the user for information and creates a option preset with the provided information.

It takes an optional argument: the URL of the application, and an option: `query` which allows limiting the detection to a given plugin.

The simplest invication is:

```
ascli config wizard
```

4.9.9 Example of configuration for a plugin

For Faspex, Shares, Node (including ATS, Aspera Transfer Service), Console, only username/password and url are required (either on command line, or from config file). Those can usually be provided on the command line:

```
ascli shares repo browse / --url=https://10.25.0.6 --username=john --password=my_password_here
```

This can also be provisioned in a config file:

- Build option preset

```
ascli config preset set shares06 url https://10.25.0.6
ascli config preset set shares06 username john
ascli config preset set shares06 password my_password_here
```

This can also be done with one single command:

```
ascli config preset init shares06
↪  @json:'{"url":"https://10.25.0.6","username":"john","password":"my_password_here"}'
```

or

```
ascli config preset update shares06 --url=https://10.25.0.6 --username=john
↪  --password=my_password_here
```

- Define this option preset as the default option preset for the specified plugin (`shares`)

```
ascli config preset set default shares shares06
```

- Display the content of configuration file in table format

```
ascli config preset overview
```

- Execute a command on the shares application using default parameters

```
ascli shares repo browse /
```

4.10 Secret Vault

Password and secrets are command options. They can be provided on command line, env vars, files etc. A more secure option is to retrieve values from a secret vault.

The vault is used with options `vault` and `vault_password`.

`vault` defines the vault to be used and shall be a `Hash`, example:

```
{"type":"system","name":"ascli"}
```

`vault_password` specifies the password for the vault. Although it can be specified on command line, for security reason you can hide the value. For example it can be securely specified on command line like this:

```
export ASCLI_VAULT_PASSWORD
read -s ASCLI_VAULT_PASSWORD
```

4.10.1 Vault: System key chain

Note: macOS only

It is possible to manage secrets in macOS key chain (only read supported currently).

```
--vault=@json:'{"type":"system","name":"ascli"}'
```

4.10.2 Vault: Encrypted file

It is possible to store and use secrets encrypted in a file.

```
--vault=@json:'{"type":"file","name":"vault.bin"}'
```

`name` is the file path, absolute or relative to the config folder `ASCLI_HOME`.

4.10.3 Vault: Operations

For this use the `config vault` command.

Then secrets can be manipulated using commands:

- `create`
- `show`
- `list`
- `delete`

```
ascli conf vault create mylabel @json:'{"password":"my_password_here","description":"for this
↪  account"}'
```

4.10.4 Configuration Finder

When a secret is needed by a sub command, the command can search for existing configurations in the config file.

The lookup is done by comparing the service URL and username (or access key).

4.10.5 Securing passwords and secrets

A passwords can be saved in clear in a option preset together with other account information (URL, username, etc...). Example:

```
ascli conf preset update myconf --url=... --username=... --password=...
```

For a more secure storage one can do:

```
ascli conf preset update myconf --url=... --username=... --password=@val:@vault:myconf.password
```

```
ascli conf vault create myconf @json:'{"password":"my_password_here"}'
```

Note: use `@val:` in front of `@vault:` so that the extended value is not evaluated.

4.11 Private Key

Some applications allow the user to be authenticated using a private key (Server, AoC, Faspex5, ...). It consists in using a pair of keys: the private key and its associated public key. The same key can be used for multiple applications. Technically, a private key contains the public key, which can be extracted from it. The file containing the private key can optionally be protected by a passphrase. If the key is protected by a passphrase, then it will be prompted when used. (some plugins support option `passphrase`)

The following commands use the shell variable `PRIVKEYFILE`. Set it to the desired safe location of the private key. Typically, located in folder `$HOME/.ssh` or `$HOME/.aspera/ascli`:

```
PRIVKEYFILE=~/.aspera/ascli/my_private_key
```

Several methods can be used to generate a key pair.

The format expected for private keys is PEM.

4.11.1 **ascli** for key generation

The generated key is of type RSA, by default: 4096 bit. For convenience, the public key is also extracted with extension `.pub`. The key is not passphrase protected.

```
ascli config genkey ${PRIVKEYFILE} 4096
```

Note: `ascli` uses the `openssl` library.

To display the version of openssl used in `ascli`:

```
ascli config echo @ruby:OpenSSL::OPENSSL_VERSION
```

4.11.2 **ssh-keygen**

Both private and public keys are generated, option `-N` is for passphrase.

```
ssh-keygen -t rsa -b 4096 -m PEM -N '' -f ${PRIVKEYFILE}
```

4.11.3 **openssl**

To generate a private key with a passphrase the following can be used on any system:

```
openssl genrsa -passout pass:_passphrase_here_ -out ${PRIVKEYFILE} 4096
openssl rsa -pubout -in ${PRIVKEYFILE} -out ${PRIVKEYFILE}.pub
```

`openssl` is sometimes compiled to support option `-nodes` (no DES, i.e. no passphrase, e.g. on macOS). In that case, add option `-nodes` instead of `-passout pass:_passphrase_here_` to generate a key without passphrase.

If option `-nodes` is not available, the passphrase can be removed using this method:

```
openssl rsa -passin pass:_passphrase_here_ -in ${PRIVKEYFILE} -out ${PRIVKEYFILE}.no_des
mv ${PRIVKEYFILE}.no_des ${PRIVKEYFILE}
```

To change (or add) the passphrase for a key do:

```
openssl rsa -des3 -in ${PRIVKEYFILE} -out ${PRIVKEYFILE}.with_des
mv ${PRIVKEYFILE}.with_des ${PRIVKEYFILE}
```

4.12 SSL CA certificate bundle

To display trusted certificate store locations:

```
ascli --show-config --fields=cert_stores
```

To modify the locations of certificate store, use option `cert_stores`. If you use this option, then default locations are removed, but they can be added using special value `DEF`. The value can be either an `Array`, or successive options.

`ascli` uses the Ruby `openssl` gem, which uses the `openssl` library. Certificates are checked against the Ruby default certificate store `OpenSSL::X509::DEFAULT_CERT_FILE` and `OpenSSL::X509::DEFAULT_CERT_DIR`, which are typically the ones of `openssl` on Unix-like systems (Linux, macOS, etc..). Ruby's default values can be overridden using env vars: `SSL_CERT_FILE` and `SSL_CERT_DIR`.

Note: One can display those values like this:

```
ascli conf echo @ruby:OpenSSL::X509::DEFAULT_CERT_DIR --format=text
ascli conf echo @ruby:OpenSSL::X509::DEFAULT_CERT_FILE --format=text
```

`ascp` also needs to validate certificates when using WSS.

Note: `ascli` overrides the default hardcoded location used by `ascp` for WSS (e.g. on macOS: `/Library/Aspera/ssl`) and uses the same locations as specified in `cert_stores` (using `-i` switch of `ascp`). Hardcoded locations can be found using:

```
strings $(ascli conf ascp info --fields=ascp)|grep -w OPENSSLDIR
```

or

```
ascli conf ascp info --fields=openssldir
```

To update trusted root certificates for `ascli`: Display the trusted certificate store locations used by `ascli`. Typically done by updating the system's root certificate store.

An up-to-date version of the certificate bundle can be retrieved with:

```
ascli conf echo @uri:https://curl.haxx.se/ca/cacert.pem --format=text
```

To download that certificate store:

```
ascli conf echo @uri:https://curl.haxx.se/ca/cacert.pem --format=text > /tmp/cacert.pem
```

Then, use this store by setting the option “ or env var export SSL_CERT_FILE

To trust a certificate (e.g. self-signed), provided that the `CN` is correct, save the certificate to a file:

```
ascli conf remote_certificate https://localhost:9092 > myserver.pem
```

Note: the saved certificateb shows the CN as first line.

Then, use this file as certificate store (e.g. here, Node API):

```
ascli conf echo @uri:https://localhost:9092/ping --cert-stores=myserver.pem
```

4.13 Image and video thumbnails

`ascli` can display thumbnails for images and videos in the terminal. This is available with the `thumbnail` command of `node` when using gen4/access key API. It's also available when using the `show` command of `preview` plugin.

The following options can be specified in the option `query`:

option	description
text	display text instead of image (Bool)
double	display double text resolution (half characters) (Bool)
font_ratio	Font height/width ratio interminal (Float)

4.14 Graphical Interactions: Browser and Text Editor

Some actions may require the use of a graphical tool:

- a browser for Aspera on Cloud authentication (web auth method)
- a text editor for configuration file edition

By default `ascli` assumes that a graphical environment is available on Windows, and on other systems, rely on the presence of the `DISPLAY` environment variable. It is also possible to force the graphical mode with option --ui :

- `--ui=graphical` forces a graphical environment, a browser will be opened for URLs or a text editor for file edition.
- `--ui=text` forces a text environment, the URL or file path to open is displayed on terminal.

4.15 Logging, Debugging

The gem is equipped with traces, mainly for debugging and learning APIs. By default logging level is `warn` and the output channel is `stderr`. To increase debug level, use parameter `log_level` (e.g. using command line `--log-level=xx`, env var `ASCLI_LOG_LEVEL`, or a parameter in the configuration file).

It is also possible to activate traces before log facility initialization using env var `ASCLI_LOG_LEVEL`.

By default passwords and secrets are removed from logs. Use option `log_secrets` set to `yes` to reveal secrets in logs.

Available loggers: `stdout`, `stderr`, `syslog`.

Available levels: `debug`, `info`, `warn`, `error`.

> Note: When using the `direct` agent (`ascp`), additional transfer logs can be activated using `ascp` options and `ascp_args`, see `direct`.

Examples:

- display debugging log on `stdout`:

```
ascli conf pre over --log-level=debug --logger=stdout
```

- log errors to `syslog`:

```
ascli conf pre over --log-level=error --logger=syslog
```

When `ascli` is used interactively in a shell, the shell itself will usually log executed commands in the history file.

4.16 Learning Aspera Product APIs (REST)

`ascli` uses mainly Aspera applications REST APIs. To display HTTP calls, use argument `-r` or `--rest-debug`, this is useful to display exact content of HTTP requests and responses.

In order to get traces of execution, use argument : `--log-level=debug`

4.17 HTTP socket parameters

To ignore SSL certificate for any address/port, use option: `insecure`, i.e. `--insecure=yes`. To ignore SSL certificate for specific address/port, use option `ignore_certificate`, set to an `Array` of URL for which certificate will be ignored (only the address and port are matched), e.g. `--ignore-certificate=@list:,https://127.0.0.1:9092`

> Note: Ignoring certificate also applies to `ascp`'s wss.

HTTP connection parameters (not `ascp` wss) can be adjusted using option `http_options`:

parameter	default
`read_timeout`	60
`write_timeout`	60
`open_timeout`	60

parameter	default
keep_alive_timeout	2

Values are in set seconds and can be of type either integer or float. Default values are the ones of Ruby: For a full list, refer to the Ruby library: Net::HTTP.

Like any other option, those can be set either on command line, or in config file, either in a global preset or server-specific one.

Example:

```
ascli aoc admin res package list --http-options=@json:'{"read_timeout":10.0}'
```

4.18 Proxy

There are several types of network connections, each of them use a different mechanism to define a (forward) proxy:

- Ruby HTTP: REST and HTTPGW client
- Legacy Aspera HTTP/S Fallback and ascp wss
- Aspera FASP

Refer to the following sections.

4.19 Proxy for REST and HTTPGW

There are two possibilities to define an HTTP proxy to be used when Ruby HTTP is used.

The `http_proxy` environment variable (lower case, preferred) can be set to the URL of the proxy. E.g. `http://myproxy.org.net:3128`. Refer to Ruby find proxy.

> Note: Ruby expects a URL and `myproxy.org.net:3128` alone is not accepted.

```
export http_proxy=http://proxy.example.com:3128
```

The `fpac` option (function for proxy auto config) can be set to a Proxy Auto Configuration (PAC) javascript value. To read the script from a URL (`http:`, `https:` and `file:`), use prefix: `@uri:`. A minimal script can be specified to define the use of a local proxy:

```
ascli --fpac='function FindProxyForURL(url, host){return "PROXY localhost:3128"}' ...
```

The result of a PAC file can be tested with command: `config proxy_check`. Example, using command line option:

```
ascli conf proxy_check --fpac='function FindProxyForURL(url, host) {return "PROXY
↪  proxy.example.com:3128;DIRECT";}' http://example.com
```

```
PROXY proxy.example.com:1234;DIRECT
```

```
ascli config proxy_check --fpac=@file:./proxy.pac http://www.example.com
```

```
PROXY proxy.example.com:8080
```

```
ascli config proxy_check --fpac=@uri:http://server/proxy.pac http://www.example.com
```

```
PROXY proxy.example.com:8080
```

If the proxy requires credentials, then use option `proxy_credentials` with username and password provided as an `Array`:

```
ascli --proxy-credentials=@json:'["__username_here__","__password_here__"]' ...
```

```
ascli --proxy-credentials=@list:__username_here__:__password_here__ ...
```

4.20 Proxy for Legacy Aspera HTTP/S Fallback

Only supported with the `direct` agent: To specify a proxy for legacy HTTP fallback, use `ascp` native option `-x` and `ascp_args`: `--transfer-info=@json:'{"ascp_args":["-x","url_here"]}'`. Alternatively, set the transfer-spec parameter: `EX_http_proxy_url`.

4.21 FASP proxy (forward) for transfers

To specify a FASP proxy (forward), set the transfer-spec parameter: `proxy` (only supported with the `direct` agent).

4.22 FASP configuration

The `config` plugin also allows specification for the use of a local FASP client. It provides the following commands for `ascp` subcommand:

- `show` : shows the path of `ascp` used
- `use` : list,download connect client versions available on internet
- `products` : list Aspera transfer products available locally
- `connect` : list,download connect client versions available on internet

4.22.1 Show path of currently used **`ascp`**

```
ascli config ascp show
```

```
/Users/laurent/.aspera/ascli/sdk/ascp
```

```
ascli config ascp info
```

```
+--------------------+-------------------------------------------------------------+
| key                | value                                                       |
+--------------------+-------------------------------------------------------------+
| ascp               | /Users/laurent/.aspera/ascli/sdk/ascp                       |
...
```

4.22.2 Selection of **`ascp`** location for **`direct`** agent

By default, `ascli` uses any found local product with `ascp`, including SDK.

To temporarily use an alternate `ascp` path use option `ascp_path` (`--ascp-path=`)

For a permanent change, the command `config ascp use` sets the same parameter for the global default.

Using a POSIX shell:

```
ascli config ascp use @path:'~/Applications/Aspera CLI/bin/ascp'
```

```
ascp version: 4.0.0.182279
Updated: global_common_defaults: ascp_path <- /Users/laurent/Applications/Aspera CLI/bin/ascp
Saved to default global preset global_common_defaults
```

Windows:

```
ascli config ascp use C:\Users\admin\.aspera\ascli\sdk\ascp.exe
```

```
ascp version: 4.0.0.182279
Updated: global_common_defaults: ascp_path <- C:\Users\admin\.aspera\ascli\sdk\ascp.exe
Saved to default global preset global_common_defaults
```

If the path has spaces, read section: Shell and Command line parsing.

4.22.3 List locally installed Aspera Transfer products

Locally installed Aspera products can be listed with:

```
ascli config ascp products list
```

```
+-------------------------------------+----------------------------------------+
| name                                | app_root                               |
+-------------------------------------+----------------------------------------+
| IBM Aspera SDK                      | /Users/laurent/.aspera/ascli/sdk       |
| Aspera Connect                      | /Applications/Aspera Connect.app       |
| IBM Aspera CLI                      | /Users/laurent/Applications/Aspera CLI |
| IBM Aspera High-Speed Transfer Server | /Library/Aspera                      |
+-------------------------------------+----------------------------------------+
```

4.22.4 Selection of local client for **ascp** for **direct** agent

If no `ascp` is selected, this is equivalent to using option: `--use-product=FIRST`.

Using the option use_product finds the `ascp` binary of the selected product.

To permanently use the `ascp` of a product:

```
ascli config ascp products use 'Aspera Connect'
saved to default global preset /Users/laurent/Applications/Aspera
 ↪  Connect.app/Contents/Resources/ascp
```

4.22.5 Installation of Connect Client on command line

```
ascli config ascp connect list
```

```
+--------------------------------------------+--------------------------------------+--------
 ↪  ---+
| id                                         | title                                | version
 ↪  |
+--------------------------------------------+--------------------------------------+--------
 ↪  ---+
| urn:uuid:589F9EE5-0489-4F73-9982-A612FAC70C4E | Aspera Connect for Windows           |
 ↪  3.11.2.63 |
| urn:uuid:A3820D20-083E-11E2-892E-0800200C9A66 | Aspera Connect for Windows 64-bit    |
 ↪  3.11.2.63 |
| urn:uuid:589F9EE5-0489-4F73-9982-A612FAC70C4E | Aspera Connect for Windows XP        |
 ↪  3.11.2.63 |
| urn:uuid:55425020-083E-11E2-892E-0800200C9A66 | Aspera Connect for Windows XP 64-bit |
 ↪  3.11.2.63 |
| urn:uuid:D8629AD2-6898-4811-A46F-2AF386531BFF | Aspera Connect for Mac Intel         |
 ↪  3.11.2.63 |
| urn:uuid:97F94DF0-22B1-11E2-81C1-0800200C9A66 | Aspera Connect for Linux 64          |
 ↪  3.11.2.63 |
+--------------------------------------------+--------------------------------------+--------
 ↪  ---+
```

```
ascli config ascp connect version 'Aspera Connect for Mac Intel' list
```

```
+-----------------------------------------+-------------------------+-----------------------
 ↪  ------------------------------------------------------------------+----------+-----------
 ↪  ------+
| title                                   | type                    | href
 ↪  | hreflang | rel                |
+-----------------------------------------+-------------------------+-----------------------
 ↪  ------------------------------------------------------------------+----------+-----------
 ↪  ------+
| Mac Intel Installer                     | application/octet-stream |
 ↪  bin/IBMAsperaConnectInstaller-3.11.2.63.dmg                                    | en
 ↪  | enclosure          |
```

```
| Mac Intel Installer                       | application/octet-stream |
 ↪  bin/IBMAsperaConnectInstallerOneClick-3.11.2.63.dmg                                  | en
 ↪  | enclosure-one-click |
| Aspera Connect for Mac HTML Documentation | text/html                |
 ↪  https://www.ibm.com/docs/en/aspera-connect/3.11?topic=aspera-connect-user-guide-macos   | en
 ↪  | documentation       |
| Aspera Connect for Mac Release Notes      | text/html                |
 ↪  https://www.ibm.com/docs/en/aspera-connect/3.11?topic=notes-release-aspera-connect-3112 | en
 ↪  | release-notes       |
+-------------------------------------------+--------------------------+-----------------------
 ↪  ------------------------------------------------------------------+----------+--------------
 ↪  ------+
```

```
ascli config ascp connect version 'Aspera Connect for Mac Intel' download enclosure --to-folder=.
```

```
Time: 00:00:02 ============================================ 100% 27766 KB/sec Time: 00:00:02
Downloaded: IBMAsperaConnectInstaller-3.11.2.63.dmg
```

4.23 Transfer Clients: Agents

Some of the actions on Aspera Applications lead to file transfers (upload and download) using the FASP protocol (`ascp`).

When a transfer needs to be started, a transfer-spec has been internally prepared. This transfer-spec will be executed by a transfer client, here called Transfer Agent.

There are currently 3 agents, set with option `transfer`:

- `direct` : a local execution of `ascp`
- `connect` : use of a local Connect Client
- `node` : use of an Aspera Transfer Node (potentially remote).
- `httpgw` : use of an Aspera HTTP Gateway
- `trsdk` : use of Aspera Transfer SDK

  Note: All transfer operations are seen from the point of view of the agent. For example, a node agent executing an ”upload”, or ”package send” operation will effectively push files to the related server from the agent node.

`ascli` standardizes on the use of a transfer-spec instead of native `ascp` options to provide parameters for a transfer session, as a common method for those three Transfer Agents.

Specific options for agents are provided with option `transfer_info`, cumulatively.

4.23.1 Direct

The `direct` agent directly executes a local `ascp`. This is the default agent for `ascli`. This is equivalent to option `--transfer=direct`. `ascli` will search locally installed Aspera products, including SDK, and use `ascp` from that component. Refer to section FASP.

The `transfer_info` option accepts the following optional parameters to control multi-session, Web Socket Session and Resume policy:

Name	Type	Description
wss	Bool	Web Socket SessionEnable use of web socket session in case it is availableDefault: true
ascp_args	Array	Array of strings with native ascp argumentsUse this instead of deprecated EX_ascp_args.Default: []
spawn_timeout_sec	Float	Multi sessionVerification time that ascp is runningDefault: 3

Name	Type	Description
spawn_delay_sec	Float	Multi sessionDelay between startup of sessionsDefault: 2
multi_incr_udp	Bool	Multi SessionIncrement UDP port on multi-sessionIf true, each session will have a different UDP port starting at `fasp_port` (or default 33001)Else, each session will use `fasp_port` (or `ascp` default)Default: true
resume	Hash	ResumeparametersSee below
resume.iter_max	int	ResumeMax number of retry on errorDefault: 7
resume.sleep_initial	int	ResumeFirst Sleep before retryDefault: 2
resume.sleep_factor	int	ResumeMultiplier of sleep period between attemptsDefault: 2
resume.sleep_max	int	ResumeDefault: 60

In case of transfer interruption, the agent will resume a transfer up to `iter_max` time. Sleep between iterations is:

```
max( sleep_max , sleep_initial * sleep_factor ^ (iter_index-1) )
```

Some transfer errors are considered ”retryable” (e.g. timeout) and some other not (e.g. wrong password). The list of known protocol errors and retry level can be listed:

```
ascli config ascp errors
```

Examples:

```
ascli ... --transfer-info=@json:'{"wss":true,"resume":{"iter_max":20}}'
ascli ... --transfer-info=@json:'{"spawn_delay_sec":2.5,"multi_incr_udp":false}'
```

> Note: The `direct` agent supports additional `transfer_spec` parameters starting with EX_ (extended). But it is preferred to use the option `transfer_info` with parameter `ascp_args`.

This can be useful to activate logging using option `-L` of `ascp`. For example, to activate debug level 2 for `ascp` (`DD`), and display those logs on the terminal (`-`):

```
--transfer-info=@json:'{"ascp_args":["-DDL-"]}'
```

This is useful to debug if a transfer fails.

To store `ascp` logs in file `aspera-scp-transfer.log` in a folder, use `--transfer-info=@json:'{"ascp_args":["-L","/path/to/folder"]}'`.

> Note: When transfer agent `direct` is used, the list of files to transfer is provided to `ascp` using either `--file-list` or `--file-pair-list` and a file list (or pair) file generated in a temporary folder. (unless `--file-list` or `--file-pair-list` is provided using `transfer_info` parameter `ascp_args`).

In addition to standard methods described in section File List, it is possible to specify the list of file using those additional methods:

- Using the pseudo transfer-spec parameter `EX_file_list`

```
--sources=@ts --ts=@json:'{"EX_file_list":"file_list.txt"}'
```

- Using option `transfer_info` parameter `ascp_args`

```
--sources=@ts --transfer-info=@json:'{"ascp_args":["--file-list","myfilelist"]}'
```

> Note: File lists is shown here, there are also similar options for file pair lists.

> Note: Those 2 additional methods avoid the creation of a copy of the file list: if the standard options `--sources=@lines:@file:... --src-type=...` are used, then the file is list read and parsed, and a new file list is created in a temporary folder.

Note: Those methods have limitations: they apply only to the `direct` transfer agent (i.e. local `ascp`) and not for Aspera on Cloud.

This agent supports a local configuration file: `aspera.conf` where Virtual links can be configured:

On a server (HSTS), the following commands can be used to set a global virtual link:

```
asconfigurator -x 'set_trunk_data;id,1;trunk_name,in;trunk_capacity,45000;trunk_on,true'
asconfigurator -x 'set_trunk_data;id,2;trunk_name,out;trunk_capacity,45000;trunk_on,true'
asconfigurator -x 'set_node_data;transfer_in_bandwidth_aggregate_trunk_id,1'
asconfigurator -x 'set_node_data;transfer_out_bandwidth_aggregate_trunk_id,2'
```

But this command is not available on clients, so edit the file `aspera.conf`, you can find the location with: `ascli conf ascp info --fields=aspera_conf` and modify the sections `default` and `trunks` like this for a global 100 Mbps virtual link:

```
<?xml version='1.0' encoding='UTF-8'?>
<CONF version="2">
    <default>
        <transfer>
            <in>
                <bandwidth>
                    <aggregate>
                        <trunk_id>1</trunk_id>
                    </aggregate>
                </bandwidth>
            </in>
            <out>
                <bandwidth>
                    <aggregate>
                        <trunk_id>2</trunk_id>
                    </aggregate>
                </bandwidth>
            </out>
        </transfer>
    </default>
    <trunks>
        <trunk>
            <id>1</id>
            <name>in</name>
            <on>true</on>
            <capacity>
                <schedule format="ranges">1000000</schedule>
            </capacity>
        </trunk>
        <trunk>
            <id>2</id>
            <name>out</name>
            <capacity>
                <schedule format="ranges">1000000</schedule>
            </capacity>
            <on>true</on>
        </trunk>
    </trunks>
</CONF>
```

It is also possible to set a schedule with different time and days, for example for the value of `schedule`:

```
start=08 end=19 days=mon,tue,wed,thu capacity=900000;1000000
```

4.23.2 IBM Aspera Connect Client GUI

By specifying option: `--transfer=connect`, `ascli` will start transfers using the locally installed Aspera Connect Client. There are no option for `transfer_info`.

4.23.3 Aspera Node API : Node to node transfers

By specifying option: `--transfer=node`, `ascli` starts transfers in an Aspera Transfer Server using the Node API, either on a local or remote node. Parameters provided in option `transfer_info` are:

Name	Type	Description
url	string	URL of the node APIMandatory
username	string	node api user or access keyMandatory
password	string	password, secret or bearer tokenMandatory
root_id	string	password or secretMandatory only for bearer token

Like any other option, `transfer_info` can get its value from a pre-configured option preset :

```
--transfer-info=@preset:_name_here_
```

or be specified using the extended value syntax :

```
--transfer-info=@json:'{"url":"https://...","username":"_user_here_","password":
↪  "my_password_here"}'
```

If `transfer_info` is not specified and a default node has been configured (name in `node` for section `default`) then this node is used by default.

If the `password` value begins with `Bearer` then the `username` is expected to be an access key and the parameter `root_id` is mandatory and specifies the root file id on the node. It can be either the access key's root file id, or any authorized file id underneath it.

4.23.4 HTTP Gateway

If it possible to send using a HTTP gateway, in case use of FASP is not allowed.

Parameters provided in option `transfer_info` are:

Name	Type	Description
url	string	URL of the HTTP GWMandatory
upload_chunk_size	int	Size in bytes of chunks for uploadDefault: 64000
api_version	string	v1 or v2, for force use of versionDefault: v2
synchronous	bool	wait for each message acknowledgmentDefault: false

Example:

```
ascli faspex package recv 323 --transfer=httpgw
↪  --transfer-info=@json:'{"url":"https://asperagw.example.com:9443/aspera/http-gwy"}'
```

Note: The gateway only supports transfers authorized with a token.

4.23.5 Transfer SDK

Another possibility is to use the Transfer SDK daemon (`asperatransferd`). Set option `transfer` to `trsdk`.

Options for `transfer_info` are:

Name	Type	Description
address	string	IP address listened by the daemonMandatoryDefault: 127.0.0.1
port	int	port of the daemonMandatoryDefault: 55002

The gem `grpc` was removed from dependencies, as it requires compilation of a native part. So, to use the Transfer SDK you should install this gem:

```
gem install grpc
```

On Windows the compilation may fail for various reasons (3.1.1):

- `cannot find -lx64-ucrt-ruby310`

  → copy the file `[Ruby main dir]\lib\libx64-ucrt-ruby310.dll.a` to `[Ruby main dir]\lib\libx64-ucrt-ruby310.a` (remove the dll extension)

- `conflicting types for 'gettimeofday'`

  → edit the file `[Ruby main dir]/include/ruby-[version]/ruby/win32.h` and change the signature of `gettimeofday` to `gettimeofday(struct timeval *, void *)` ,i.e. change `struct timezone` to `void`

4.24 Transfer Specification

Some commands lead to file transfer (upload/download). All parameters necessary for this transfer are described in a transfer-spec (Transfer Specification), such as:

- server address
- transfer user name
- credentials
- file list
- etc...

`ascli` builds the transfer-spec internally as a `Hash`. It is not necessary to provide additional parameters on the command line for this transfer.

It is possible to modify or add any of the supported transfer-spec parameter using the `ts` option. The `ts` option accepts a `Hash` Extended Value containing one or several transfer-spec parameters. Multiple `ts` options on command line are cumulative, and the `Hash` value is deeply merged. To remove a (deep) key from transfer spec, set the value to `null`.

> Note: Default transfer spec values can be displayed with command: `config ascp info --flat-hash=no` under field `ts`.

It is possible to specify `ascp` options when the `transfer` option is set to `direct` using `transfer_info` option parameter: `ascp_args`. Example: `--transfer-info=@json:'{"ascp_args":["-l","100m"]}'`. This is especially useful for `ascp` command line parameters not supported in the transfer spec.

The use of a transfer-spec instead of `ascp` parameters has the advantage of:

- common to all Transfer Agent
- not dependent on command line limitations (special characters...)

4.25 Transfer Parameters

All standard transfer-spec parameters can be specified. transfer-spec can also be saved/overridden in the config file.

References:

- Aspera Node API Documentation → /opt/transfers
- Aspera Transfer SDK Documentation → Guides → API Ref → Transfer Spec V1
- Aspera Connect SDK → search `The parameters for starting a transfer.`

Parameters can be displayed with commands:

```
ascli config ascp spec
ascli config ascp spec --select=@json:'{"d":"Y"}' --fields=-d,n,c
```

Columns:

- D=Direct (local `ascp` execution)
- N=Node API
- C=Connect Client
- T=Transfer SDK
- H=HTTP Gateway

`ascp` argument or environment variable is provided in description.

Fields with EX_ prefix are extensions to transfer agent `direct`. (only in `ascli`).

Field	Type	D	N	C	T	H	Description
apply_local_doc-root	bool	Y					Apply local docroot to source paths.(--apply-local-docroot)
authentication	string			Y			value=token for SSH bypass keys, else password asked if not provided.()
cipher	string	Y	Y	Y	Y	Y	In transit encryption type.Allowed values: none, aes-128, aes-192, aes-256, aes-128-cfb, aes-192-cfb, aes-256-cfb, aes-128-gcm, aes-192-gcm, aes-256-gcm(-c (conversion){enum})
cipher_allowed	string	Y	Y	Y	Y	Y	returned by node API. Valid literals include ”aes-128” and ”none”.()
compression	int						ascp4 only, 0 / 1?()
content_protection	string	Y	Y	Y	Y	Y	Enable client-side encryption at rest. (CSEAR, content protection)Allowed values: encrypt, decrypt(--file-crypt {enum})
content_protec-tion_password	string	Y	Y	Y	Y	Y	Specifies CSEAR password. (content protection)(env:ASPERA_SCP_FILEPASS)
cookie	string	Y	Y	Y	Y	Y	Metadata for transfer specified by application(env:ASPERA_SCP_COOKIE)
create_dir	bool	Y	Y	Y	Y	Y	Specifies whether to create new directories.(-d)
delete_be-fore_transfer	bool	Y	Y	Y	Y	Y	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 objectson the source are not deleted.(--delete-before-transfer)
delete_source	bool	Y	Y				Remove SRC files after transfer success(--remove-after-transfer)
destination_root	string	Y	Y	Y	Y	Y	Destination root directory.()
destination_root_id	string						The file ID of the destination root directory.Required when using Bearer token auth for the destination node.()
dgram_size	int	Y	Y	Y	Y	Y	UDP datagram size in bytes(-Z {int})
direction	string	Y	Y	Y	Y	Y	Direction of transfer (on client side)Allowed values: send, receive(--mode (conversion){enum})
ex-clude_newer_than	int	Y					skip src files with mtime > arg(--exclude-newer-than {int})
exclude_older_than	int	Y					skip src files with mtime < arg(--exclude-older-than {int})
fasp_port	int	Y	Y	Y	Y	Y	Specifies fasp (UDP) port.(-O {int})
fasp_url	string						Only used in Faspex.()
file_checksum	string	Y	Y				Enable checksum reporting for transferred files by specifying the hash to use.Allowed values: sha-512, sha-384, sha-256, sha1, md5, none()
http_fallback	boolstring	Y	Y	Y	Y	Y	When true(1), attempts to perform an HTTP transfer if a FASP transfer cannot be performed.(-y (conversion){bool}
http_fallback_port	int	Y					Specifies http port when no cipher is used(-t {int})

Field	Type	D	N	C	T	H	Description
https_fallback_port	int	Y	Y	Y	Y	Y	Specifies https port when cipher is used(-t {int})
keepalive	bool	Y					The session is running in persistent session mode.(--keepalive)
lock_min_rate	bool	Y	Y	Y	Y	Y	TODO: remove ?()
lock_min_rate_kbps	bool			Y			If true, lock the minimum transfer rate to the value set for min_rate_kbps.If false, users can adjust the transfer rate up to the value set for target_rate_cap_kbps.()
lock_rate_policy	bool			Y			If true, lock the rate policy to the default value.()
lock_target_rate	bool	Y	Y	Y	Y	Y	TODO: remove ?()
lock_target_rate_kbps	bool	Y	Y	Y	Y	Y	If true, lock the target transfer rate to the default value set for target_rate_kbps.If false, users can adjust the transfer rate up to the value set for target_rate_cap_kbps.()
min_rate_cap_kbps	int	Y	Y	Y	Y	Y	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. (Default: 0)()
min_rate_kbps	int	Y	Y	Y	Y	Y	Set the minimum transfer rate in kilobits per second.(-m {int})
move_after_transfer	string	Y	Y				The relative path to which the files will be moved after the transfer at the source side. Available as of 3.8.0.(--move-after-transfer {string})
multi_session	int	Y	Y	Y	Y	Y	Use multi-session transfer. max 128.Each participant on one host needs an independent UDP (-O) port.Large files are split between sessions only when transferring with resume_policy=none.()
multi_session_threshold	int	Y	Y				Split files across multiple ascp sessions if their size in bytes is greater than or equal to the specified value.(0=no file is split)(--multi-session-threshold {int})
obfuscate_file_names	bool					Y	HTTP Gateway obfuscates file names when set to true.()
overwrite	string	Y	Y	Y	Y	Y	Overwrite destination files with the source files of the same name.Allowed values: never, always, diff, older, diff+older(--overwrite {enum})
password	string		Y				Password for local Windows user when transfer user associated with node api user is not the same as the one running asperanoded.Allows impersonating the transfer user and have access to resources (e.g. network shares).Windows only, node api only.()
paths	array	Y	Y	Y	Y	Y	Array of path to the source (required) and a path to the destination (optional).()
precalculate_job_size	bool	Y	Y	Y	Y	Y	Specifies whether to precalculate the job size.(--precalculate-job-size)

Field	Type	D	N	C	T	H	Description
preserve_access_time	bool	Y	Y	Y	Y	Y	Preserve the source-file access timestamps at the destination.Because source access times are updated by the transfer operation,the timestamp that is preserved is the one just before to the transfer.(--preserve-access-time)
preserve_acls	string	Y					Preserve access control lists.Allowed values: none, native, metafile(--preserve-acls {enum})
preserve_creation_time	bool	Y	Y	Y	Y	Y	(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-creation-time)
preserve_extended_attrs	string						Preserve the extended attributes.Allowed values: none, native, metafile(--preserve-xattrs {enum})
preserve_file_owner_gid	bool	Y					Preserve the group ID for a file owner(--preserve-file-owner-gid)
preserve_file_owner_uid	bool	Y					Preserve the user ID for a file owner(--preserve-file-owner-uid)
preserve_modification_time	bool	Y	Y	Y	Y	Y	Set the modification time, the last time a file or directory was modified (written), of a transferred fileto the modification of the source file or directory.Preserve source-file modification timestamps at the destination.(--preserve-modification-time)
preserve_remote_acls	string	Y					Preserve remote access control lists.Allowed values: none, native, metafile(--remote-preserve-acls {enum})
preserve_source_access_time	bool	Y					Preserve the time logged for when the source file was accessed(--preserve-source-access-time)
preserve_times	bool		Y				Preserve file timestamps.(--preserve-times)
proxy	string	Y					Specify the address of the Aspera high-speed proxy server.dnat(s)://[user[:password]@]server:portDefault ports for DNAT and DNATS protocols are 9091 and 9092.Password, if specified here, overrides the value of environment variable ASPERA_PROXY_PASS.(--proxy {string})
rate_policy	string	Y	Y	Y	Y	Y	The transfer rate policy to use when sharing bandwidth.Allowed values: low, fair, high, fixed(--policy {enum})
rate_policy_allowed	string			Y			Specifies most aggressive rate policy that is allowed.Returned by node API.Allowed values: low, fair, high, fixed()
read_threads	int						ascp4 only()
remote_access_key	string						The access key ID of the access key that was used to construct the bearer token that is used to authenticate to the remote node.()
remote_host	string	Y	Y	Y	Y	Y	IP or fully qualified domain name of the remote server(--host {string})
remote_password	string	Y	Y	Y	Y	Y	SSH session password(env:ASPERA_SCP_PASS)

Field	Type	D	N	C	T	H	Description
remote_user	string	Y	Y	Y	Y	Y	Remote user. Default value is ”xfer” on node or connect.(--user {string})
remove_after_transfer	bool	Y	Y				Remove SRC files after transfer success(--remove-after-transfer)
remove_empty_directories	bool	Y	Y				Specifies whether to remove empty directories.(--remove-empty-directories)
remove_empty_source_directory	bool	Y					Remove empty source subdirectories and remove the source directory itself, if empty(--remove-empty-source-directory)
remove_skipped	bool	Y	Y	Y			Must also have remove_after_transfer set to true, Defaults to false, if true, skipped files will be removed as well.(--remove-skipped)
resume_policy	string	Y	Y	Y	Y	Y	If a transfer is interrupted or fails to finish, resume without re-transferring the whole files.Allowed values: none, attrs, sparse_csum, full_csum(-k (conversion){enum})
retry_duration	stringint		Y	Y			Specifies how long to wait before retrying transfer. (e.g. ”5min”)()
source_root	string	Y	Y	Y	Y	Y	Path to be prepended to each source path.This is either a conventional path or it can be a URI but only if there is no root defined.(--source-prefix64 (conversion){string})
source_root_id	string		Y				The file ID of the source root directory. Required when using Bearer token auth for the source node.()
src_base	string	Y	Y				Specify the prefix to be stripped off from each source object.The remaining portion of the source path is kept intact at the destination.Special care must be taken when used with cloud storage.(--src-base64 (conversion){string})
ssh_args	string						Array of arguments to pass to SSH. Use with caution.()
ssh_port	int	Y	Y	Y	Y	Y	Specifies SSH (TCP) port. Default: local:22, other:33001(-P {int})
ssh_private_key	string	Y					Private key used for SSH authentication.Shall look like: -----BEGIN RSA PRIV4TE KEY-----\nMII...Note the JSON encoding: \n for newlines.(env:ASPERA_SCP_KEY)
ssh_private_key_passphrase	string	Y					The passphrase associated with the transfer user's SSH private key. Available as of 3.7.2.(env:ASPERA_SCP_PASS)
sshfp	string	Y	Y	Y	Y	Y	Check it against server SSH host key fingerprint(--check-sshfp {string})
symlink_policy	string	Y	Y	Y	Y	Y	Handle source side symbolic linksAllowed values: follow, copy, copy+force, skip(--symbolic-links {enum})
tags	hash	Y	Y	Y	Y	Y	Metadata for transfer as JSON(--tags64 (conversion){hash})
target_rate_cap_kbps	int			Y			Returned by upload/download_setup node API.()
target_rate_kbps	int	Y	Y	Y	Y	Y	Specifies desired speed for the transfer.(-l {int})

Field	Type	D	N	C	T	H	Description
target_rate_percentage	string	Y	Y	Y	Y	Y	TODO: remove ?()
title	string		Y	Y			Title of the transfer()
token	string	Y	Y	Y	Y	Y	Authorization token: Bearer, Basic or ATM (Also arg -W)(env:ASPERA_SCP_TOKEN)
use_ascp4	bool	Y	Y				specify version of protocol()
use_system_ssh	string						TODO, comment...()
write_threads	int						ascp4 only()
wss_enabled	bool	Y	Y	Y	Y	Y	Server has Web Socket service enabled()
wss_port	int	Y	Y	Y	Y	Y	TCP port used for websocket service feed()
EX_ascp_args	array	Y					DEPRECATED: (4.13) Use option transfer_info.ascp_argsAdd native command line arguments to ascp()
EX_at_rest_password	string	Y					DEPRECATED: (4.13) Use standard spec parameter: content_protection_passwordContent protection password(env:ASPERA_SCP_FILEPASS)
EX_file_list	string	Y					DEPRECATED: (4.14) Use command line file list, or option transfer_info.ascp_argssource file list()
EX_file_pair_list	string	Y					DEPRECATED: (4.14) Use command line file pair list, or option transfer_info.ascp_argssource file pair list()
EX_http_proxy_url	string	Y					DEPRECATED: (4.14) TODO, use proxy option ?Specify the proxy server address used by HTTP Fallback(-x {string})
EX_http_transfer_jpeg	int	Y					DEPRECATED: (4.14) Use option transfer_info.ascp_argsHTTP transfers as JPEG file(-j {int})
EX_license_text	string	Y					DEPRECATED: (4.14) Use env var ASPERA_SCP_LICENSELicense file text override.By default ascp looks for license file near executable.(env:ASPERA_SCP_LICENSE)
EX_no_read	bool	Y					DEPRECATED: (4.14) Use option transfer_info.ascp_argsno read source(--no-read)
EX_no_write	bool	Y					DEPRECATED: (4.14) Use option transfer_info.ascp_argsno write on destination(--no-write)
EX_proxy_password	string	Y					DEPRECATED: (4.14) Use env var ASPERA_PROXY_PASSPassword used for Aspera proxy server authentication.May be overridden by password in URL provided in parameter: proxy.(env:ASPERA_PROXY_PASS)
EX_ssh_key_paths	array	Y					DEPRECATED: (4.14) Use option transfer_info.ascp_argsUse public key authentication for SSH and specify the private key file paths(-i {array})

4.25.1 Destination folder for transfers

The destination folder is set by `ascli` by default to:

- `.` for downloads

- / for uploads

It is specified by the transfer-spec parameter `destination_root`. As such, it can be modified with option: `--ts=@json:'{"destination_root":"<path>"}'`. The option `to_folder` provides an equivalent and convenient way to change this parameter: `--to-folder=<path>`.

4.25.2 List of files for transfers

When uploading, downloading or sending files, the user must specify the list of files to transfer.

By default the list of files to transfer is simply provided on the command line.

The list of (source) files to transfer is specified by (extended value) option `sources` (default: `@args`). The list is either simply the list of source files, or a combined source/destination list (see below) depending on value of option `src_type` (default: `list`).

In `ascli`, all transfer parameters, including file list, are provided to the transfer agent in a transfer-spec so that execution of a transfer is independent of the transfer agent (direct, connect, node, transfer sdk...). So, eventually, the list of files to transfer is provided to the transfer agent using the transfer-spec field: `"paths"` which is a list (array) of pairs of `"source"` (mandatory) and `"destination"` (optional). The `sources` and `src_type` options provide convenient ways to populate the transfer spec with the source file list.

Possible values for option `sources` are:

- `@args` : (default) the list of files (or file pair) is directly provided on the command line (after commands): unused arguments (not starting with `-`) are considered as source files. So, by default, the list of files to transfer will be simply specified on the command line. Example:

  ```
  ascli server upload ~/first.file secondfile
  ```

  This is the same as (with default values):

  ```
  ascli server upload --sources=@args --src-type=list ~/mysample.file secondfile
  ```

- an Extended Value with type Array of String

  Note: extended values can be tested with the command `conf echo`

  Examples:

  - Using extended value

    Create the file list:

    ```
    echo ~/mysample.file > myfilelist.txt
    echo secondfile >> myfilelist.txt
    ```

    Use the file list: one path per line:

    ```
    --sources=@lines:@file:myfilelist.txt
    ```

  - Using JSON array

    ```
    --sources=@json:'["file1","file2"]'
    ```

  - Using STDIN, one path per line

    ```
    --sources=@lines:@stdin:
    ```

  - Using Ruby code (one path per line in file)

    ```
    --sources=@ruby:'File.read("myfilelist.txt").split("\n")'
    ```

- `@ts` : the user provides the list of files directly in the `paths` field of transfer spec (option `ts`). Examples:

  - Using transfer spec

  ```
  --sources=@ts --ts=@json:'{"paths":[{"source":"file1"},{"source":"file2"}]}'
  ```

The option `src_type` allows specifying if the list specified in option `sources` is a simple file list or if it is a file pair list.

Note: Option `src_type` is not used if option `sources` is set to `@ts`

Supported values for `src_type` are:

- `list` : (default) the path of destination is the same as source and each entry is a source file path
- `pair` : the first element is the first source, the second element is the first destination, and so on.

Example: Source file `200KB.1` is renamed `sample1` on destination:

```
ascli server upload --src-type=pair ~/Documents/Samples/200KB.1 /Upload/sample1
```

> Note: There are some specific rules to specify a file list when using Aspera on Cloud, refer to the AoC plugin section.

4.25.3 Source directory structure on destination

This section is not specific to `ascli` it is `ascp` behaviour.

The transfer destination is normally expected to designate a destination folder.

But there is one exception: The destination specifies the new item name when the following are met:

- there is a single source item (file or folder)
- transfer spec `create_dir` is not set to `true` (`ascp` option `-d` not provided)
- destination is not an existing folder
- the `dirname` of destination is an existing folder

For this reason it is recommended to set `create_dir` to `true` for consistent behaviour between single and multiple items transfer, this is the default in `ascli`.

If a simple source file list is provided (no `destination` in `paths`, i.e. no `file_pair_list` provided), the destination folder is used as destination folder for each source file, and source file folder names are not preserved.

The inner structure of source items that are folder is preserved on destination.

A leading / on destination is ignored (relative to docroot) unless docroot is not set (relative to home).

In the following table source folder `d3` contains 2 files: `f1` and `d4/f2`.

Source files	Destination	Folders on Dest.	`create_dir`	Destination Files
f1	d/f	-	false	Error: `d` does not exist.
f1	d/f	d	false	d/f (renamed)
f1	d/f/.	d	false	d/f (renamed)
f1	d/f	d/f	false	d/f/f1
f1 f2	d	d	false	d/f1 d/f2
d3	d	-	false	d/f1 d/f2 (renamed)
f1	d	-	true	d/f1
f1 f2	d	-	true	d/f1 d/f2
d1/f1 d2/f2	d	-	true	d/f1 d/f2
d3	d	-	true	d/d3/f1 d/d3/d4/f2

If a file par list is provided then it is possible to rename or specify a different destination folder for each source (relative to the destination).

If transfer spec has a `src_base`, it has the side effect that the simple source file list is considered as a file pair list, and so the lower structure of source folders is preserved on destination.

Source files	Destination	`src_base`	Destination Files
d1/d2/f2 d1/d3/f3	d	d1	d/d2/f2 d/d3/f3

Advanced Example: Send files `./file1` and `./folder2/files2` to server (e.g. `/Upload`) and keep the original file names and folders, i.e. send `file1` to `/Upload/file1` and `files2` to `/Upload/folder2/files2`.

- If files are specified as `./file1 ./folder2/files2`,

  then destination will be: `/Upload/file1 /Upload/files2`

- One possibility is to specify a file pair list: `--src-type=pair file1 file1 folder2/files2 folder2/files2`
- Another possibility is to specify a source base: `--src-base=$PWD $PWD/file1 $PWD/folder2/files2`

  (note that `.` cannot be used as source base)
- Similarly, create a temporary soft link (Linux): `ln -s . tmp_base` and use `--src-base=tmp_base tmp_base/file1 tmp_base/folder2/files2`
- One can also similarly use `--sources=@ts` and specify the list of files in the `paths` field of transfer spec with both `source` and `destination` for each file.

4.25.4 Support of multi-session

Multi session, i.e. starting a transfer of a file set using multiple sessions (one `ascp` process per session) is supported on `direct` and `node` agents, not yet on connect.

- `--transfer=node`

```
--ts=@json:'{"multi_session":10,"multi_session_threshold":1}'
```

Multi-session is directly supported by the node daemon.

- `--transfer=direct`

```
--ts=@json:'{"multi_session":5,"multi_session_threshold":1,"resume_policy":"none"}'
```

  Note: `resume_policy` set to `attr` may cause problems: `none` or `sparse_csum` shall be preferred.

`ascli` starts multiple `ascp` for Multi-session using `direct` agent.

When multi-session is used, one separate UDP port is used per session (refer to `ascp` manual page).

4.25.5 Content protection

Also known as Client-side encryption at rest (CSEAR), content protection allows a client to send files to a server which will store them encrypted (upload), and decrypt files as they are being downloaded from a server, both using a passphrase, only known by users sharing files. Files stay encrypted on server side.

Activating CSEAR consists in using transfer spec parameters:

- `content_protection` : activate encryption (`encrypt` for upload) or decryption (`decrypt` for download)
- `content_protection_password` : the passphrase to be used.

Example: parameter to download a faspex package and decrypt on the fly

```
--ts=@json:'{"content_protection":"decrypt","content_protection_password":"my_password_here"}'
```

4.25.6 Transfer Spec Examples

- Change target rate

```
--ts=@json:'{"target_rate_kbps":500000}'
```

- Override the FASP SSH port to a specific TCP port:

```
--ts=@json:'{"ssh_port":33002}'
```

- Force http fallback mode:

```
--ts=@json:'{"http_fallback":"force"}'
```

- Activate progress when not activated by default on server

```
--ts=@json:'{"precalculate_job_size":true}'
```

4.26 Transfer progress bar

Control with option `progressbar` (`Bool`), by default it is `yes` if the output is a terminal.

To display the native progress bar of `ascp`, use `--progressbar=no --transfer-info=@json:'{"quiet":false}'`.

4.27 Scheduler

It is useful to configure automated scheduled execution. `ascli` does not provide an internal scheduler. Instead, use the service provided by the Operating system:

4.27.1 Windows Scheduler

Windows provides the Task Scheduler. It can be configured:

- Using utility `schtasks.exe`
- Using powershell function scheduletasks
- Using `taskschd.msc` (UI)

4.27.2 Unix-like Scheduler

Unix-like systems (Linux, ...) provide cron, configured using a crontab

Linux also provides `anacron`, if tasks are hourly or daily.

For example, on Linux it is convenient to create a wrapping script, e.g. `cron_ascli` that will setup the environment (e.g. Ruby) to properly start `ascli`:

```
#!/bin/bash
# load the ruby environment
. /etc/profile.d/rvm.sh
rvm use 2.6 --quiet
# set a timeout protection, just in case ascli is frozen
tmout=30m
# forward arguments to ascli
exec timeout ${tmout} ascli "${@}"
```

Example of cronjob created for user `xfer`.

```
crontab<<EOF
0    * * * *  /home/xfer/cron_ascli preview scan --logger=syslog --display=error
2-59 * * * *  /home/xfer/cron_ascli preview trev --logger=syslog --display=error
EOF
```

> Note: The logging options are kept here in the cronfile instead of conf file to allow execution on command line with output on command line.

4.28 Locking for exclusive execution

In some cases one needs to ensure that `ascli` is not executed several times in parallel.

When `ascli` is executed automatically on a schedule basis, one generally desires that a new execution is not started if a previous execution is still running because an on-going operation may last longer than the scheduling period:

- Executing instances may pile-up and kill the system
- The same file may be transferred by multiple instances at the same time.
- `preview` may generate the same files in multiple instances.

Usually the OS native scheduler already provides some sort of protection against parallel execution:

- The Windows scheduler does this by default
- Linux cron can leverage the utility `flock` to do the same:

```
/usr/bin/flock -w 0 /var/cron.lock ascli ...
```

`ascli` natively supports a locking mechanism with option `lock_port`. (Technically, this opens a local TCP server port, and fails if this port is already used, providing a local lock. Lock is released when process exits).

Testing `ascli` locking:

Run this same command in two separate terminals within less than 30 seconds:

```
ascli config echo @ruby:'sleep(30)' --lock-port=12345
```

The first instance will sleep 30 seconds, the second one will immediately exit like this:

```
WARN -- : Another instance is already running (Address already in use - bind(2) for "127.0.0.1"
 ↪  port 12345).
```

4.29 ”Provençale”

`ascp`, the underlying executable implementing Aspera file transfer using FASP, has a capability to not only access the local file system (using system's `open`,`read`,`write`,`close` primitives), but also to do the same operations on other data storage such as S3, Hadoop and others. This mechanism is call PVCL. Several PVCL adapters are available, some are embedded in `ascp` , some are provided om shared libraries and must be activated. (e.g. using `trapd`)

The list of supported PVCL adapters can be retrieved with command:

```
ascli conf ascp info
```

```
+--------------------+-------------------------------------------------------------+
| key                | value                                                       |
+--------------------+-------------------------------------------------------------+
-----8<-----snip-----8<-----
| product_name       | IBM Aspera SDK                                              |
| product_version    | 4.0.1.182389                                                |
| process            | pvcl                                                        |
| shares             | pvcl                                                        |
| noded              | pvcl                                                        |
| faux               | pvcl                                                        |
| file               | pvcl                                                        |
| stdio              | pvcl                                                        |
| stdio-tar          | pvcl                                                        |
+--------------------+-------------------------------------------------------------+
```

Here we can see the adapters: `process`, `shares`, `noded`, `faux`, `file`, `stdio`, `stdio-tar`.

Those adapters can be used wherever a file path is used in `ascp` including configuration. They act as a pseudo ”drive”.

The simplified format is:

```
<adapter>:///<sub file path>?<arg1>=<val1>&...
```

One of the adapters, used in this manual, for testing, is `faux`. It is a pseudo file system allowing generation of file data without actual storage (on source or destination).

4.30 **`faux:`** for testing

This is an extract of the man page of `ascp`. This feature is a feature of `ascp`, not `ascli`.

This adapter can be used to simulate a file or a directory.

To discard data at the destination, the destination argument is set to `faux://`.

To send uninitialized data in place of an actual source file, the source file is replaced with an argument of the form:

```
faux:///filename?filesize
```

where:

- `filename` is the name that will be assigned to the file on the destination
- `filesize` is the number of bytes that will be sent (in decimal).

Note: characters ? and & are shell special characters (wildcard and background), so `faux` file specification on command line should be protected (using quotes or \). If not, the shell may give error: `no matches found` or equivalent.

For all sizes, a suffix can be added (case insensitive) to the size: k,m,g,t,p,e (values are power of 2, e.g. 1M is 220, i.e. 1 mebibyte, not megabyte). The maximum allowed value is 8*260. Very large `faux` file sizes (petabyte range and above) will likely fail due to lack of destination storage unless destination is `faux://`.

To send uninitialized data in place of a source directory, the source argument is replaced with an argument of the form:

```
faux:///dirname?<arg1>=<val1>&...
```

where:

- `dirname` is the folder name and can contain / to specify a subfolder.
- supported arguments are:

Name	Type	Description
count	int	mandatory
file	string	Basename for filesDefault: ”file”
size	int	Size of first file.Default: 0
inc	int	Increment applied to determine next file sizeDefault: 0
seq	enum	Sequence in determining next file sizeValues: random, sequentialDefault: sequential
buf_init	enum	How source data is initializedOption 'none' is not allowed for downloads.Values:none, zero, randomDefault:zero

The sequence parameter is applied as follows:

- If `seq` is `random` then each file size is:
  - size +/- (inc * rand())
  - Where rand is a random number between 0 and 1
  - Note that file size must not be negative, inc will be set to size if it is greater than size
  - Similarly, overall file size must be less than 8260. If size + inc is greater, inc will be reduced to limit size + inc to 7260.
- If `seq` is `sequential` then each file size is:
  - `size + ((file_index - 1) * inc)`
  - Where first file is index 1
  - So file1 is `size` bytes, file2 is `size + inc` bytes, file3 is `size + inc * 2` bytes, etc.
  - As with `random`, `inc` will be adjusted if `size + (count * inc)` is not less then 8*260.

Filenames generated are of the form: `<file>_<00000 ... count>_<filesize>`

Examples:

- Upload 20 gibibytes of random data to file myfile to directory /Upload

```
ascli server upload faux:///myfile\?20g --to-folder=/Upload
```

- Upload a file /tmp/sample but do not save results to disk (no docroot on destination)

```
ascli server upload /tmp/sample --to-folder=faux://
```

- Upload a faux directory `mydir` containing 1 million files, sequentially with sizes ranging from 0 to 2 Mebibyte - 2 bytes, with the basename of each file being `testfile` to /Upload

```
ascli server upload "faux:///mydir?file=testfile&count=1m&size=0&inc=2&seq=sequential"
 ↪  --to-folder=/Upload
```

4.31 Usage

```
ascli -h
ERROR: Other(ArgumentError): wrong number of arguments (given 3, expected 0)
ERROR: Argument: unprocessed options: ["-h"]
Use --log-level=debug to get more details.
```

Note: commands and parameter values can be written in short form.

4.32 Bulk creation and deletion of resources

Bulk creation and deletion of resources are possible using option `bulk` (`yes`,`no`(default)). In that case, the operation expects an `Array` of `Hash` instead of a simple `Hash` using the Extended Value Syntax. This option is available only for some of the resources: if you need it: try and see if the entities you try to create or delete support this option.

4.33 Plugins

`ascli` uses a plugin mechanism. The first level command (just after `ascli` on the command line) is the name of the concerned plugin which will execute the command. Each plugin usually represents commands sent to a specific application. For instance, the plugin `faspex` allows operations on the application "Aspera Faspex".

Available plugins can be found using command:

```
ascli conf plugin list
```

```
+--------------+--------+--------+-----------------------------------------------------+
| plugin       | detect | wizard | path                                                |
+--------------+--------+--------+-----------------------------------------------------+
| shares       | Y      | Y      | .../aspera-cli/lib/aspera/cli/plugins/shares.rb     |
| node         | Y      | Y      | .../aspera-cli/lib/aspera/cli/plugins/node.rb       |
...
+--------------+--------+--------+-----------------------------------------------------+
```

Most plugins will take the URL option: `url` to identify their location.

REST APIs of Aspera legacy applications (Aspera Node, Faspex 4, Shares, Console, Orchestrator) use simple username/password authentication: HTTP Basic Authentication using options: `username` and `password`.

Aspera on Cloud and Faspex 5 rely on Oauth.

By default plugins are looked-up in folders specified by (multi-value) option `plugin_folder`:

```
ascli --show-config --select=@json:'{"key":"plugin_folder"}'
```

You can create the skeleton of a new plugin like this:

```
ascli conf plugin create foo .
```

```
Created ./foo.rb
```

```
ascli --plugin-folder=. foo
```

Chapter 5

Plugin: **aoc**: IBM Aspera on Cloud

Aspera on Cloud API requires the use of Oauth v2 mechanism for authentication (HTTP Basic authentication is not supported).

It is recommended to use the wizard to set it up, although manual configuration is also possible.

5.1 Configuration: using Wizard

`ascli` provides a configuration wizard.

The wizard guides you through the steps to create a new configuration preset for Aspera on Cloud.

The first

Here is a sample invocation :

```
ascli config wizard
option: url> https://myorg.ibmaspera.com
Detected: Aspera on Cloud
Preparing preset: aoc_myorg
Please provide path to your private RSA key, or empty to generate one:
option: pkeypath>
using existing key:
/Users/myself/.aspera/ascli/aspera_aoc_key
Using global client_id.
option: username> john@example.com
Updating profile with new key
creating new config preset: aoc_myorg
Setting config preset as default for aspera
saving config file
Done.
You can test with:
ascli aoc user profile show
```

Optionally, it is possible to create a new organization-specific ”integration”, i.e. client application identification. For this, specify the option: `--use-generic-client=no`.

If you already know the application, and want to limit the detection to it, provide url and plugin name:

```
ascli config wizard myorg aoc
```

5.2 Configuration: using manual setup

Note: If you used the wizard (recommended): skip this section.

5.2.1 Configuration details

Several types of OAuth authentication are supported:

- JSON Web Token (JWT) : authentication is secured by a private key (recommended for `ascli`)
- Web based authentication : authentication is made by user using a browser
- URL Token : external users authentication with url tokens (public links)

The authentication method is controlled by option `auth`.

For a quick start, follow the mandatory and sufficient section: API Client Registration (auth=web) as well as [option preset](#lprt) for Aspera on Cloud.

For a more convenient, browser-less, experience follow the JWT section (auth=jwt) in addition to Client Registration.

In Oauth, a "Bearer" token is generated to authenticate REST calls. Bearer tokens are valid for a period of time defined (by the AoC app, configurable by admin) at its creation. `ascli` saves generated tokens in its configuration folder, tries to re-use them or regenerates them when they have expired.

5.2.2 Optional: API Client Registration

If you use the built-in client_id and client_secret, skip this and do not set them in next section.

Else you can use a specific OAuth API client_id, the first step is to declare `ascli` in Aspera on Cloud using the admin interface.

(AoC documentation: Registering an API Client).

Let's start by a registration with web based authentication (auth=web):

- Open a web browser, log to your instance: e.g. `https://myorg.ibmaspera.com/`
- Go to Apps → Admin → Organization → Integrations
- Click "Create New"
  - Client Name: `ascli`
  - Redirect URIs: `http://localhost:12345`
  - Origins: `localhost`
  - uncheck "Prompt users to allow client to access"
  - leave the JWT part for now
- Save

  Note: for web based authentication, `ascli` listens on a local port (e.g. specified by the redirect_uri, in this example: 12345), and the browser will provide the OAuth code there. For 'ascli, HTTP is required, and 12345 is the default port.

Once the client is registered, a "Client ID" and "Secret" are created, these values will be used in the next step.

5.2.3 option preset for Aspera on Cloud

If you did not use the wizard, you can also manually create a option preset for `ascli` in its configuration file.

Lets create an option preset called: `my_aoc_org` using `ask` interactive input (client info from previous step):

```
ascli config preset ask my_aoc_org url client_id client_secret
option: url> https://myorg.ibmaspera.com/
option: client_id> my_client_id_here
option: client_secret> my_client_secret_here
updated: my_aoc_org
```

(This can also be done in one line using the command `config preset update my_aoc_org --url=...`)

Define this option preset as default configuration for the `aspera` plugin:

```
ascli config preset set default aoc my_aoc_org
```

Note: Default `auth` method is `web` and default `redirect_uri` is `http://localhost:12345`. Leave those default values.

5.2.4 Activation of JSON Web Token (JWT) for direct authentication

For a Browser-less, Private Key-based authentication, use the following steps.

In order to use JWT for Aspera on Cloud API client authentication, a private/public key pair must be used.

5.2.4.1 API Client JWT activation

If you are not using the built-in client_id and secret, JWT needs to be authorized in Aspera on Cloud. This can be done in two manners:

- Graphically
  - Open a web browser, log to your instance: `https://myorg.ibmaspera.com/`
  - Go to Apps → Admin → Organization → Integrations
  - Click on the previously created application
  - select tab : ”JSON Web Token Auth”
  - Modify options if necessary, for instance: activate both options in section ”Settings”
  - Click ”Save”
- Using command line

```
ascli aoc admin res client list
```

```
+------------+---------------+
|     id     |  name         |
+------------+---------------+
| my_BJbQiFw | my-client-app |
+------------+---------------+
```

```
ascli aoc admin res client modify my_BJbQiFw
 ↪  @json:'{"jwt_grant_enabled":true,"explicit_authorization_required":false}'
```

```
modified
```

5.2.5 User key registration

The public key must be assigned to your user. This can be done in two manners:

5.2.5.1 Graphically

Open the previously generated public key located here: `$HOME/.aspera/ascli/my_private_key.pub`

- Open a web browser, log to your instance: `https://myorg.ibmaspera.com/`
- Click on the user's icon (top right)
- Select ”Account Settings”
- Paste the Public Key in the ”Public Key” section
- Click on ”Submit”

5.2.5.2 Using command line

```
ascli aoc admin res user list
```

```
+--------+----------------+
|   id   |      name      |
+--------+----------------+
| 109952 | Tech Support   |
| 109951 | LAURENT MARTIN |
+--------+----------------+
```

```
ascli aoc user profile modify
 ↪  @ruby:'{"public_key"=>File.read(File.expand_path("~/.aspera/ascli/my_private_key.pub"))}'
```

```
modified
```

Note: the `aspera user info show` command can be used to verify modifications.

5.2.6 option preset modification for JWT

To activate default use of JWT authentication for `ascli` using the option preset, do the following:

- change auth method to JWT
- provide location of private key
- provide username to login as (OAuth ”subject”)

Execute:

```
ascli config preset update my_aoc_org --auth=jwt
↪  --private-key=@val:@file:~/.aspera/ascli/my_private_key --username=someuser@example.com
```

> Note: the private key argument represents the actual PEM string. In order to read the content from a file, use the `@file:` prefix. But if the @file: argument is used as is, it will read the file and set in the config file. So to keep the ”@file” tag in the configuration file, the `@val:` prefix is added.

After this last step, commands do not require web login anymore.

5.2.7 Public and private links

AoC gives the possibility to generate public links for both the `Files` and `Packages` modules. Public links embed the authorization of access. Provide the public link using option `url` alone.

In addition, the `Files` application supports private links. Private links require the user to authenticate. So, provide the same options as for regular authentication, and provide the private link using option `url`.

A user may not be part of any workspace, but still have access to shared folders (using private links). In that case, it is possible to list those shared folder by using a value for option `workspace` equal to `@none:` or `@json:null` or `@ruby:nil`.

5.2.8 First Use

Once client has been registered and option preset created: `ascli` can be used:

```
ascli aoc files br /
```

```
Current Workspace: Default Workspace (default)
empty
```

5.3 Calling AoC APIs from command line

The command `ascli aoc bearer` can be used to generate an OAuth token suitable to call any AoC API (use the `scope` option to change the scope, default is `user:all`). This can be useful when a command is not yet available.

Example:

```
curl -s -H "Authorization: $(ascli aoc bearer_token)"
↪  'https://api.ibmaspera.com/api/v1/group_memberships?embed[]=dropbox&embed[]=workspace'|jq -r
↪  '.[]|(.workspace.name + " -> " + .dropbox.name)'
```

It is also possible to get the bearer token for node, as user or as admin using:

```
ascli aoc files bearer_token_node /
```

```
ascli aoc admin res node v4 1234 --secret=_ak_secret_here_ bearer_token_node /
```

5.4 Administration

The `admin` command allows several administrative tasks (and require admin privilege).

It allows actions (create, update, delete) on ”resources”: users, group, nodes, workspace, etc... with the `admin resource` command.

5.4.1 Listing resources

The command `aoc admin res <type> list` lists all entities of given type. It uses paging and multiple requests if necessary.

The option `query` can be optionally used. It expects a `Hash` using Extended Value Syntax, generally provided using: `--query=@json:{...}`. Values are directly sent to the API call and used as a filter on server side.

The following parameters are supported:

- `q` : a filter on name of resource (case insensitive, matches if value is contained in name)
- `sort`: name of fields to sort results, prefix with `-` for reverse order.
- `max` : maximum number of items to retrieve (stop pages when the maximum is passed)
- `pmax` : maximum number of pages to request (stop pages when the maximum is passed)
- `page` : native api parameter, in general do not use (added by
- `per_page` : native api parameter, number of items par api call, in general do not use
- Other specific parameters depending on resource type.

Both `max` and `pmax` are processed internally in `ascli`, not included in actual API call and limit the number of successive pages requested to API. `ascli` will return all values using paging if not provided.

Other parameters are directly sent as parameters to the GET request on API.

`page` and `per_page` are normally added by `ascli` to build successive API calls to get all values if there are more than 1000. (AoC allows a maximum page size of 1000).

`q` and `sort` are available on most resource types.

Other parameters depend on the type of entity (refer to AoC API).

Examples:

- List users with `laurent` in name:

```
ascli aoc admin res user list --query=@json:'{"q":"laurent"}'
```

- List users who logged-in before a date:

```
ascli aoc admin res user list --query=@json:'{"q":"last_login_at:<2018-05-28"}'
```

- List external users and sort in reverse alphabetical order using name:

```
ascli aoc admin res user list --query=@json:'{"member_of_any_workspace":false,"sort":"-name"}'
```

Refer to the AoC API for full list of query parameters, or use the browser in developer mode with the web UI.

> Note: The option `select` can also be used to further refine selection, refer to section earlier.

5.4.2 Selecting a resource

Resources are identified by a unique `id`, as well as a unique `name` (case insensitive).

To execute an action on a specific resource, select it using one of those methods:

- recommended: give id directly on command line after the action: `aoc admin res node show 123`
- give name on command line after the action: `aoc admin res node show name abc`
- provide option `id` : `aoc admin res node show 123`
- provide option `name` : `aoc admin res node show --name=abc`

5.4.3 Creating a resource

New resources (users, groups, workspaces, etc..) can be created using a command like:

```
ascli aoc admin res create <resource type> @json:'{<...parameters...>}'
```

Some of the API endpoints are described here. Sadly, not all.

Nevertheless, it is possible to guess the structure of the creation value by simply dumping an existing resource, and use the same parameters for the creation.

```
ascli aoc admin res group show 12345 --format=json
```

```
{"created_at":"2018-07-24T21:46:39.000Z","description":null,"id":"12345","manager":false,"name":"A8Demo
↪  WS1","owner":false,"queued_operation_count":0,"running_operation_count":0,
↪  "stopped_operation_count":0,"updated_at":"2018-07-24T21:46:39.000Z","saml_group":false,
↪  "saml_group_dn":null,"system_group":true,"system_group_type":"workspace_members"}
```

Remove the parameters that are either obviously added by the system: `id`, `created_at`, `updated_at` or optional.

And then craft your command:

```
ascli aoc admin res group create @json:'{"description":"test to delete","name":"test 1 to
↪  delete","saml_group":false}'
```

If the command returns an error, example:

```
+----+-----------------------------------------------------------------------------------+
| id | status                                                                            |
+----+-----------------------------------------------------------------------------------+
|    | found unpermitted parameters: :manager, :owner, :system_group, :system_group_type |
|    | code: unpermitted_parameters                                                      |
|    | request_id: b0f45d5b-c00a-4711-acef-72b633f8a6ea                                  |
|    | api.ibmaspera.com 422 Unprocessable Entity                                        |
+----+-----------------------------------------------------------------------------------+```
```

Well, remove the offending parameters and try again.

> Note: Some properties that are shown in the web UI, such as membership, are not listed directly in the resource, but instead another resource is created to link a user and its group: `group_membership`

5.4.4 Access Key secrets

In order to access some administrative actions on nodes (in fact, access keys), the associated secret is required. The secret is provided using the `secret` option. For example in a command like:

```
ascli aoc admin res node 123 --secret="my_secret_here" v3 info
```

It is also possible to store secrets in the secret vault and then automatically find the related secret using the config finder.

5.4.5 Activity

The activity app can be queried with:

```
ascli aoc admin analytics transfers
```

It can also support filters and send notification using option `notif_to`. a template is defined using option `notif_template`:

`mytemplate.erb`:

```
From: <%=from_name%> <<%=from_email%>>
To: <<%=ev['user_email']%>>
Subject: <%=ev['files_completed']%> files received

Dear <%=ev[:user_email.to_s]%>,
We received <%=ev['files_completed']%> files for a total of <%=ev['transferred_bytes']%> bytes,
↪  starting with file:
<%=ev['content']%>

Thank you.
```

The environment provided contains the following additional variable:

- ev : all details on the transfer event

Example:

```
ascli aoc admin analytics transfers --once-only=yes --lock-port=12345
↪  --query=@json:'{"status":"completed","direction":"receive"}' --notif-to=active
↪  --notif-template=@file:mytemplate.erb
```

Options:

- `once_only` keep track of last date it was called, so next call will get only new events
- `query` filter (on API call)
- `notify` send an email as specified by template, this could be places in a file with the `@file` modifier.

  Note: This must not be executed in less than 5 minutes because the analytics interface accepts only a period of time between 5 minutes and 6 months. The period is [date of previous execution]..[now].

5.4.6 Transfer: Using specific transfer ports

By default transfer nodes are expected to use ports TCP/UDP 33001. The web UI enforces that. The option `default_ports` ([yes]/no) allows ascli to retrieve the server ports from an API call (download_setup) which reads the information from `aspera.conf` on the server.

5.4.7 Using ATS

Refer to section "Examples" of ATS and substitute command `ats` with `aoc admin ats`.

5.4.8 Files with type **`link`**

Aspera on Cloud Shared folders are implemented through a special type of file: `link`. A `link` is the equivalent of a symbolic link on a file system: it points to another folder (not file).

Listing a link (in terminal position of path) will information on the link itself, not the content of the folder it points to. To list the target folder content, add a `/` a the end of the path.

Example:

```
$ ascli aoc files br the_link
Current Workspace: Default (default)
+------------+------+----------------+------+----------------------+--------------+
| name       | type | recursive_size | size | modified_time        | access_level |
+------------+------+----------------+------+----------------------+--------------+
| the_link   | link |                |      | 2021-04-28T09:17:14Z | edit         |
+------------+------+----------------+------+----------------------+--------------+
```

```
$ ascli aoc files br the_link/
Current Workspace: Default (default)
+-------------+------+----------------+------+----------------------+--------------+
| name        | type | recursive_size | size | modified_time        | access_level |
+-------------+------+----------------+------+----------------------+--------------+
| file_inside | file |                |      | 2021-04-26T09:00:00Z | edit         |
+-------------+------+----------------+------+----------------------+--------------+
```

5.4.9 Example: Bulk creation of users

```
ascli aoc admin res user create --bulk=yes
↪  @json:'[{"email":"dummyuser1@example.com"},{"email":"dummyuser2@example.com"}]'
```

```
+-------+---------+
|  id   | status  |
+-------+---------+
| 98398 | created |
| 98399 | created |
+-------+---------+
```

5.4.10 Example: Find with filter and delete

```
ascli aoc admin res user list --query='@json:{"q":"dummyuser"}' --fields=id,email
```

```
+-------+------------------------+
|  id   |         email          |
+-------+------------------------+
| 98398 | dummyuser1@example.com |
| 98399 | dummyuser2@example.com |
+-------+------------------------+
```

```
ascli aoc admin res user list --query='@json:{"q":"dummyuser"}' --fields=id --display=data
 ↪  --format=csv | ascli aoc admin res user delete @lines:@stdin: --bulk=yes
```

```
+-------+---------+
|  id   | status  |
+-------+---------+
| 98398 | deleted |
| 98399 | deleted |
+-------+---------+
```

5.4.11 Example: Find deactivated users since more than 2 years

```
ascli aoc admin res user list --query=@ruby:'{"deactivated"=>true,"q"=>"last_login_at:
 ↪  <#{(DateTime.now.to_time.utc-2*365*86400).iso8601}"}'
```

To delete them use the same method as before

5.4.12 Example: Display current user's workspaces

```
ascli aoc user workspaces list
```

```
+------+----------------------------+
|  id  |            name            |
+------+----------------------------+
| 16   | Engineering                |
| 17   | Marketing                  |
| 18   | Sales                      |
+------+----------------------------+
```

5.4.13 Example: Create a sub access key in a ”node”

Creation of a sub-access key is like creation of access key with the following difference: authentication to node API is made with accesskey (master access key) and only the path parameter is provided: it is relative to the storage root of the master key. (id and secret are optional)

```
ascli aoc admin resource node --name=_node_name_ --secret=_secret_ v4 access_key create
 ↪  @json:'{"storage":{"path":"/folder1"}}'
```

5.4.14 Example: Display transfer events (ops/transfer)

```
ascli aoc admin res node --secret=_secret_ v3 transfer list
 ↪  --query=@json:'[["q","*"],["count",5]]'
```

Examples of query:

```
{"q":"type(file_upload OR file_delete OR file_download OR file_rename OR folder_create OR
 ↪  folder_delete OR folder_share OR folder_share_via_public_link)","sort":"-date"}
```

```
{"tag":"aspera.files.package_id=LA8OU3p8w"}
```

5.4.15 Example: Display node events (events)

```
ascli aoc admin res node --secret=_secret_ v3 events
```

5.4.16 Example: Display members of a workspace

```
ascli aoc admin res workspace_membership list --fields=member_type,manager,member.email
 ↪  --query=@json:'{"embed":"member","inherited":false,"workspace_id":11363,"sort":"name"}'
```

```
+-------------+---------+-----------------------------------+
| member_type | manager |           member.email            |
+-------------+---------+-----------------------------------+
| user        | true    | john.curtis@email.com             |
| user        | false   | someuser@example.com |
| user        | false   | jean.dupont@me.com                |
| user        | false   | another.user@example.com          |
| group       | false   |                                   |
| user        | false   | aspera.user@gmail.com             |
+-------------+---------+-----------------------------------+
```

Other query parameters:

```
{"workspace_membership_through":true,"include_indirect":true}
```

5.4.17 Example: add all members of a workspace to another workspace

a- Get id of first workspace

```
WS1='First Workspace'
WS1ID=$(ascli aoc admin res workspace list --query=@json:'{"q":"'"$WS1"'"}'
 ↪  --select=@json:'{"name":"'"$WS1"'"}' --fields=id --format=csv)
```

b- Get id of second workspace

```
WS2='Second Workspace'
WS2ID=$(ascli aoc admin res workspace list --query=@json:'{"q":"'"$WS2"'"}'
 ↪  --select=@json:'{"name":"'"$WS2"'"}' --fields=id --format=csv)
```

c- Extract membership information

```
ascli aoc admin res workspace_membership list --fields=manager,member_id,member_type,workspace_id
 ↪  --query=@json:'{"workspace_id":'"$WS1ID"'}' --format=jsonpp > ws1_members.json
```

d- Convert to creation data for second workspace:

```
grep -Eve '(direct|effective_manager|_count|storage|"id")' ws1_members.json|sed '/workspace_id/
 ↪  s/"'"$WS1ID"'"/"'"$WS2ID"'"/g' > ws2_members.json
```

or, using jq:

```
jq '[.[] | {member_type,member_id,workspace_id,manager,workspace_id:"'"$WS2ID"'"}]'
 ↪  ws1_members.json > ws2_members.json
```

e- Add members to second workspace

```
ascli aoc admin res workspace_membership create --bulk=yes @json:@file:ws2_members.json
```

5.4.18 Example: Get users who did not log since a date

```
ascli aoc admin res user list --fields=email --query=@json:'{"q":"last_login_at:<2018-05-28"}'
```

```
+------------------------------+
|             email            |
+------------------------------+
| John.curtis@acme.com         |
| Jean.Dupont@tropfort.com     |
+------------------------------+
```

5.4.19 Example: List ”Limited” users

```
ascli aoc admin res user list --fields=email --select=@json:'{"member_of_any_workspace":false}'
```

5.4.20 Example: create a group, add to workspace and add user to group

- Create the group and take note of `id`

```
ascli aoc admin res group create @json:'{"name":"group 1","description":"my super group"}'
```

Group: 11111

- Get the workspace id

```
ascli aoc admin res workspace list --query=@json:'{"q":"myworkspace"}' --fields=id --format=csv
↪  --display=data
```

Workspace: 22222

- Add group to workspace

```
ascli aoc admin res workspace_membership create
↪  @json:'{"workspace_id":22222,"member_type":"user","member_id":11111}'
```

- Get a user's id

```
ascli aoc admin res user list --query=@json:'{"q":"manu.macron@example.com"}' --fields=id
↪  --format=csv --display=data
```

User: 33333

- Add user to group

```
ascli aoc admin res group_membership create
↪  @json:'{"group_id":11111,"member_type":"user","member_id":33333}'
```

5.4.21 Example: Perform a multi Gbps transfer between two remote shared folders

In this example, a user has access to a workspace where two shared folders are located on different sites, e.g. different cloud regions.

First, setup the environment (skip if already done)

```
ascli conf wizard --url=https://sedemo.ibmaspera.com --username=someuser@example.com
```

```
Detected: Aspera on Cloud
Preparing preset: aoc_sedemo
Using existing key:
/Users/laurent/.aspera/ascli/aspera_aoc_key
Using global client_id.
Please Login to your Aspera on Cloud instance.
Navigate to your "Account Settings"
Check or update the value of "Public Key" to be:
-----BEGIN PUBLIC KEY-----
SOME PUBLIC KEY PEM DATA HERE
-----END PUBLIC KEY-----
Once updated or validated, press enter.

creating new config preset: aoc_sedemo
Setting config preset as default for aspera
saving config file
Done.
You can test with:
ascli aoc user profile show
```

This creates the option preset ”aoc_<org name>” to allow seamless command line access and sets it as default for aspera on cloud.

Then, create two shared folders located in two regions, in your files home, in a workspace.

Then, transfer between those:

```
ascli -Paoc_show aoc files transfer --from-folder='IBM Cloud SJ' --to-folder='AWS Singapore'
↪  100GB.file
↪  --ts=@json:'{"target_rate_kbps":"1000000","multi_session":10,"multi_session_threshold":1}'
```

5.4.22 Example: create registration key to register a node

```
ascli aoc admin res client create @json:'{"data":{"name":"laurentnode","client_subject_scopes":
↪  ["alee","aejd"],"client_subject_enabled":true}}' --fields=token
↪  --format=csv
```

```
jfqslfdjlfdjfhdjklqfhdkl
```

5.4.23 Example: delete all registration keys

```
ascli aoc admin res client list --fields=id --format=csv|ascli aoc admin res client delete
↪  @lines:@stdin: --bulk=yes
```

```
+-----+---------+
| id  | status  |
+-----+---------+
| 99  | deleted |
| 100 | deleted |
| 101 | deleted |
| 102 | deleted |
+-----+---------+
```

5.4.24 Example: Create a Node

AoC nodes as actually composed with two related entities:

- An access key created on the Transfer Server (HSTS/ATS)
- a `node` resource in the AoC application.

The web UI allows creation of both entities in one shot. For more flexibility, `ascli` allows this in two separate steps.

> Note: When selecting "Use existing access key" in the web UI, this actually skips access key creation (first step).

So, for example, the creation of a node using ATS in IBM Cloud looks like (see other example in this manual):

- Create the access key on ATS

  The creation options are the ones of ATS API, refer to the section on ATS for more details and examples.

  ```
  ascli aoc admin ats access_key create --cloud=softlayer --region=eu-de
  ↪  --params=@json:'{"storage":{"type":"ibm-s3","bucket":"mybucket","credentials":
  ↪  {"access_key_id":"mykey","secret_access_key":"mysecret"},"path":"/"}}'
  ```

  Once executed, the access key `id` and `secret`, randomly generated by the node api, is displayed.

  > Note: Once returned by the API, the secret will not be available anymore, so store this preciously. ATS secrets can only be reset by asking to IBM support.

- Create the AoC node entity

  First, Retrieve the ATS node address

  ```
  ascli aoc admin ats cluster show --cloud=softlayer --region=eu-de --fields=transfer_setup_url
  ↪  --format=csv --transpose-single=no
  ```

  Then use the returned address for the `url` key to actually create the AoC Node entity:

  ```
  ascli aoc admin res node create
  ↪  @json:'{"name":"myname","access_key":"myaccesskeyid","ats_access_key":true,
  ↪  "ats_storage_type":"ibm-s3","url":"https://ats-sl-fra-all.aspera.io"}'
  ```

Creation of a node with a self-managed node is similar, but the command `aoc admin ats access_key create` is replaced with `node access_key create` on the private node itself.

5.5 List of files to transfer

Source files are provided as a list with the `sources` option. Refer to section File list

> Note: A special case is when the source files are located on Aspera on Cloud (i.e. using access keys and the `file id` API).

Source files are located on "Aspera on cloud", when :

- the server is Aspera on Cloud, and executing a download or recv
- the agent is Aspera on Cloud, and executing an upload or send

In this case:

- If there is a single file : specify the full path
- Else, if there are multiple files:
  - All source files must be in the same source folder
  - Specify the source folder as first item in the list
  - followed by the list of file names.

5.6 Packages

The webmail-like application.

5.6.1 Send a Package

General syntax:

```
ascli aoc packages send [package extended value] [other parameters such as file list and transfer
↪  parameters]
```

Notes:

- Package creation parameter are sent as positional argument. Refer to the AoC package creation API, or display an existing package in JSON to list attributes.
- List allowed shared inbox destinations with: `ascli aoc packages shared_inboxes list`
- Use fields: `recipients` and/or `bcc_recipients` to provide the list of recipients: user or shared inbox.
  - Provide either ids as expected by API: `"recipients":[{"type":"dropbox","id":"1234"}]`
  - or just names: `"recipients":[{"The Dest"}]` . ascli will resolve the list of email addresses and dropbox names to the expected type/id list, based on case insensitive partial match.
- If a user recipient (email) is not already registered and the workspace allows external users, then the package is sent to an external user, and
  - if the option `new_user_option` is `@json:{"package_contact":true}` (default), then a public link is sent and the external user does not need to create an account
  - if the option `new_user_option` is `@json:{}`, then external users are invited to join the workspace

5.6.2 Example: Send a package with one file to two users, using their email

```
ascli aoc packages send @json:'{"name":"my title","note":"my
↪  note","recipients":["someuser@example.com","other@example.com"]}' my_file.dat
```

5.6.3 Example: Send a package to a shared inbox with metadata

```
ascli aoc packages send --workspace=eudemo @json:'{"name":"my pack title","recipients":["Shared
↪  Inbox With Meta"],"metadata":{"Project
↪  Id":"123","Type":"Opt2","CheckThose":["Check1","Check2"],"Optional
↪  Date":"2021-01-13T15:02:00.000Z"}}' ~/Documents/Samples/200KB.1
```

It is also possible to use identifiers and API parameters:

```
ascli aoc packages send --workspace=eudemo @json:'{"name":"my pack title","recipients":[{"type":
↪  "dropbox","id":"12345"}],"metadata":[{"input_type":"single-text","name":"Project
↪  Id","values":["123"]},{"input_type":"single-dropdown","name":"Type","values":["Opt2"]},
↪  {"input_type":"multiple-checkbox","name":"CheckThose","values":["Check1","Check2"]},
↪  {"input_type":"date","name":"Optional Date","values":["2021-01-13T15:02:00.000Z"]}]}'
↪  ~/Documents/Samples/200KB.1
```

5.6.4 Example: List packages in a given shared inbox

When user packages are listed, the following query is used:

```
{"archived":false,"exclude_dropbox_packages":true,"has_content":true,"received":true}
```

To list packages in a shared inbox, the query has to be specified with the the shared inbox by name or its identifier. Additional parameters can be specified, as supported by the API (to find out available filters, consult the API definition, or use the web interface in developer mode). The current workspace is added unless specified in the query.

Using shared inbox name:

```
ascli aoc packages list --query=@json:'{"dropbox_name":"My Shared
↪  Inbox","archived":false,"received":true,"has_content":true,"exclude_dropbox_packages":false,
↪  "include_draft":false,"sort":"-received_at"}'
```

Using shared inbox identifier: first retrieve the id of the shared inbox, and then list packages with the appropriate filter.

```
shared_box_id=$(ascli aoc packages shared_inboxes show --name='My Shared Inbox' --format=csv
↪  --display=data --fields=id --transpose-single=no)
```

```
ascli aoc packages list --query=@json:'{"dropbox_id":"'$shared_box_id'","archived":false,
↪  "received":true,"has_content":true,"exclude_dropbox_packages":false,"include_draft":false,
↪  "sort":"-received_at"}'
```

5.6.5 Example: Receive all packages from a given shared inbox

```
ascli aoc packages recv ALL --workspace=_workspace_ --once-only=yes --lock-port=12345
↪  --query=@json:'{"dropbox_name":"_shared_inbox_name_","archived":false,"received":true,
↪  "has_content":true,"exclude_dropbox_packages":false,"include_draft":false}'
↪  --ts=@json:'{"resume_policy":"sparse_csum","target_rate_kbps":50000}'
```

5.6.6 Example: Send a package with files from the Files app

Find files in Files app:

```
ascli aoc files browse /src_folder
```

```
+---------------+--------+----------------+-------------+----------------------+--------------+
| name          | type   | recursive_size | size        | modified_time        | access_level |
+---------------+--------+----------------+-------------+----------------------+--------------+
| sample_video  | link   |                |             | 2020-11-29T22:49:09Z | edit         |
| 100G          | file   |                | 107374182400 | 2021-04-21T18:19:25Z | edit         |
| 10M.dat       | file   |                | 10485760    | 2021-05-18T08:22:39Z | edit         |
| Test.pdf      | file   |                | 1265103     | 2022-06-16T12:49:55Z | edit         |
+---------------+--------+----------------+-------------+----------------------+--------------+
```

Let's send a package with the file `10M.dat` from subfolder /src_folder in a package:

```
ascli aoc files node_info /src_folder --format=json --display=data | ascli aoc packages send
↪  @json:'{"name":"test","recipients":["someuser@example.com"]}' 10M.dat --transfer=node
↪  --transfer-info=@json:@stdin:
```

5.6.7 Receive new packages only (Cargo)

It is possible to automatically download new packages, like using Aspera Cargo:

```
ascli aoc packages recv ALL --once-only=yes --lock-port=12345
```

- `ALL` (case sensitive) will download all packages

- `--once-only=yes` keeps memory of any downloaded package in persistency files located in the configuration folder
- `--lock-port=12345` ensures that only one instance is started at the same time, to avoid running two downloads in parallel

Typically, one would execute this command on a regular basis, using the method of your choice: see Scheduler.

5.7 Files

The Files application presents a Home folder to users in a given workspace. Files located here are either user's files, or shared folders.

> Note: All commands under `files` are the same as under `access_keys do self` for plugin `node`, i.e. gen4/access key operations.

5.7.1 Download Files

The general download command is:

```
ascli aoc files download <source folder path> <source filename 1> ...
```

I.e. the first argument is the source folder, and the following arguments are the source file names in this folder.

If a single file or folder is to be downloaded, then a single argument can be provided.

```
ascli aoc files download <single file path>
```

5.7.2 Shared folders

Shared folder created by users are managed through permissions.

For creation, parameters are the same as for node API permissions. `ascli` expects the same payload for creation, but it will automatically populate required tags if needed.

Also, the pseudo key `with` is available: it will lookup the name in the contacts and fill the proper type and id. The pseudo parameter `link_name` allows changing default "shared as" name.

- List permissions on a shared folder as user

```
ascli aoc files perm /shared_folder_test1 list
```

- Share a personal folder with other users

```
ascli aoc files perm /shared_folder_test1 create @json:'{"with":"laurent"}'
```

- Revoke shared access

```
ascli aoc files perm /shared_folder_test1 delete 6161
```

Public and Private short links can be managed with command:

```
ascli aoc files short_link private create _path_here_
ascli aoc files short_link private list _path_here_
ascli aoc files short_link public list _path_here_
ascli aoc files short_link public delete _id_
```

5.7.3 Cross Organization transfers

It is possible to transfer files directly between organizations without having to first download locally and then upload...

Although optional, the creation of option preset is recommended to avoid placing all parameters in the command line.

Procedure to send a file from org1 to org2:

- Get access to Organization 1 and create a option preset: e.g. `org1`, for instance, use the Wizard
- Check that access works and locate the source file e.g. `mysourcefile`, e.g. using command `files browse`
- Get access to Organization 2 and create a option preset: e.g. `org2`

- Check that access works and locate the destination folder `mydestfolder`
- execute the following:

```
ascli -Porg1 aoc files node_info /mydestfolder --format=json --display=data | ascli -Porg2 aoc
↪  files upload mysourcefile --transfer=node --transfer-info=@json:@stdin:
```

Explanation:

- `-Porg1 aoc` use Aspera on Cloud plugin and load credentials for `org1`
- `files node_info /mydestfolder` generate transfer information including node api credential and root id, suitable for the next command
- `--format=json` format the output in JSON (instead of default text table)
- `--display=data` display only the result, and remove other information, such as workspace name
- `|` the standard output of the first command is fed into the second one
- `-Porg2 aoc` use Aspera on Cloud plugin and load credentials for `org2`
- `files upload mysourcefile` upload the file named `mysourcefile` (located in `org1`)
- `--transfer=node` use transfer agent type `node` instead of default `direct`
- `--transfer-info=@json:@stdin:` provide `node` transfer agent information, i.e. node API credentials, those are expected in JSON format and read from standard input

5.7.4 Find Files

The command `aoc files find` allows to search for files in a given workspace.

It works also on `node` resource using the `v4` command:

```
ascli aoc admin res node --name='my node name' --secret='my_secret_here' v4 find ...
```

For instructions, refer to section `find` for plugin `node`.

5.8 AoC sample commands

```
aoc admin analytics transfers organization
↪  --query=@json:'{"status":"completed","direction":"receive"}' --notif-to=my_email_external
↪  --notif-template=@ruby:'%Q{From: <%=from_name%> <<%=from_email%>>\nTo: <<%=to%>>\nSubject:
↪  <%=ev["files_completed"]%> files received\n\n<%=ev.to_yaml%>}'
aoc admin ats access_key create --cloud=aws --region=my_bucket_region
↪  --params=@json:'{"id":"ak_aws","name":"my test key
↪  AWS","storage":{"type":"aws_s3","bucket":"my_bucket_name","credentials":{"access_key_id":
↪  "my_bucket_key","secret_access_key":"my_bucket_secret"},"path":"/"}}'
aoc admin ats access_key create --cloud=softlayer --region=my_bucket_region
↪  --params=@json:'{"id":"ak1ibmcloud","secret":"my_secret_here","name":"my test
↪  key","storage":{"type":"ibm-s3","bucket":"my_bucket_name","credentials":{"access_key_id":
↪  "my_bucket_key","secret_access_key":"my_bucket_secret"},"path":"/"}}'
aoc admin ats access_key delete ak1ibmcloud
aoc admin ats access_key list --fields=name,id
aoc admin ats access_key node ak1ibmcloud --secret=my_secret_here browse /
aoc admin ats cluster clouds
aoc admin ats cluster list
aoc admin ats cluster show --cloud=aws --region=eu-west-1
aoc admin ats cluster show 1f412ae7-869a-445c-9c05-02ad16813be2
aoc admin auth_providers list
aoc admin res application list
aoc admin res client list
aoc admin res client_access_key list
aoc admin res client_registration_token create @json:'{"data":{"name":"test_client_reg1",
↪  "client_subject_scopes":["alee","aejd"],"client_subject_enabled":true}}'
aoc admin res client_registration_token delete my_clt_reg_id
aoc admin res client_registration_token list
aoc admin res contact list
aoc admin res dropbox list
aoc admin res dropbox_membership list
aoc admin res group list
```

```
aoc admin res kms_profile list
aoc admin res node list
aoc admin res operation list
aoc admin res organization show
aoc admin res package list --http-options=@json:'{"read_timeout":120.0}'
aoc admin res saml_configuration list
aoc admin res self show
aoc admin res short_link list
aoc admin res user list
aoc admin res user modify %name:my_user_email @json:'{"deactivated":false}'
aoc admin res workspace_membership list
aoc admin resource node do %name:my_ak_name --secret=my_ak_secret browse /
aoc admin resource node do %name:my_ak_name --secret=my_ak_secret delete /folder1
aoc admin resource node do %name:my_ak_name --secret=my_ak_secret mkdir /folder1
aoc admin resource node do %name:my_ak_name --secret=my_ak_secret v3 access_key create
 ↪  @json:'{"id":"testsub1","storage":{"path":"/folder1"}}'
aoc admin resource node do %name:my_ak_name --secret=my_ak_secret v3 access_key delete testsub1
aoc admin resource node do %name:my_ak_name --secret=my_ak_secret v3 events
aoc admin resource workspace list
aoc admin resource workspace_membership list --fields=ALL --query=@json:'{"page":1,"per_page":
 ↪  50,"embed":"member","inherited":false,"workspace_id":11363,"sort":"name"}'
aoc admin subscription
aoc automation workflow action my_wf_id create @json:'{"name":"toto"}' \
aoc automation workflow create @json:'{"name":"test_workflow"}'
aoc automation workflow delete my_wf_id
aoc automation workflow list
aoc automation workflow list --query=@json:'{"show_org_workflows":"true"}' --scope=admin:all
aoc automation workflow list --select=@json:'{"name":"test_workflow"}' --fields=id --format=csv
 ↪  --display=data
aoc bearer_token --display=data --scope=user:all
aoc files bearer /
aoc files bearer_token_node / --cache-tokens=no
aoc files browse /
aoc files browse / --url=my_privlink
aoc files browse / --url=my_publink_folder_nopass
aoc files browse / --url=my_publink_folder_pass --password=my_publink_password
aoc files delete /testsrc
aoc files download --transfer=connect testdst/testfile.bin
aoc files find / '\.partial$'
aoc files http_node_download --to-folder=. testdst/testfile.bin
aoc files mkdir /testsrc
aoc files modify my_test_folder
aoc files permission my_test_folder list
aoc files rename /somefolder testdst
aoc files short_link private create /testdst
aoc files short_link private list /testdst
aoc files short_link public create testdst
aoc files show %id:my_file_id
aoc files show /
aoc files show testdst/testfile.bin
aoc files sync admin status --sync-info=@json:'{"name":"my_aoc_sync2","reset":true,"direction":
 ↪  "pull","local":{"path":"/data/local_sync"},"remote":{"path":"/testdst"}}'
aoc files sync admin status --sync-info=@json:'{"sessions":[{"name":"my_aoc_sync1","direction":
 ↪  "pull","local_dir":"/data/local_sync","remote_dir":"/testdst","reset":true}]}'
aoc files sync start --sync-info=@json:'{"name":"my_aoc_sync2","reset":true,"direction":"pull",
 ↪  "local":{"path":"/data/local_sync"},"remote":{"path":"/testdst"}}'
aoc files sync start --sync-info=@json:'{"sessions":[{"name":"my_aoc_sync1","direction":"pull",
 ↪  "local_dir":"/data/local_sync","remote_dir":"/testdst","reset":true}]}'
aoc files thumbnail my_test_folder/video_file.mpg
aoc files thumbnail my_test_folder/video_file.mpg --query=@json:'{"text":true,"double":true}'
aoc files transfer push /testsrc --to-folder=/testdst testfile.bin
aoc files upload --to-folder=/ testfile.bin --url=my_publink_folder_nopass
aoc files upload --to-folder=/testsrc testfile.bin
```

```
aoc files upload --workspace=my_other_workspace --to-folder=my_other_folder testfile.bin
 ↪  --transfer=node --transfer-info=@json:@stdin:
aoc files v3 info
aoc gateway https://localhost:12345/aspera/faspex
aoc org --url=my_publink_recv_from_aocuser
aoc organization
aoc packages browse "my_package_id" /contents
aoc packages list
aoc packages list --query=@json:'{"dropbox_name":"my_shbx_name","sort":"-received_at",
 ↪  "archived":false,"received":true,"has_content":true,"exclude_dropbox_packages":false}'
aoc packages recv "my_package_id" --to-folder=.
aoc packages recv ALL --to-folder=. --once-only=yes --lock-port=12345
aoc packages recv ALL --to-folder=. --once-only=yes --lock-port=12345
 ↪  --query=@json:'{"dropbox_name":"my_shbx_name","archived":false,"received":true,
 ↪  "has_content":true,"exclude_dropbox_packages":false,"include_draft":false}'
 ↪  --ts=@json:'{"resume_policy":"sparse_csum","target_rate_kbps":50000}'
aoc packages send --workspace=my_shbx_ws @json:'{"name":"Important files delivery","recipients":
 ↪  ["my_shbx_name"],"metadata":[{"input_type":"single-text","name":"Project
 ↪  Id","values":["123"]},{"input_type":"single-dropdown","name":"Type","values":["Opt2"]},
 ↪  {"input_type":"multiple-checkbox","name":"CheckThose","values":["Check1","Check2"]},
 ↪  {"input_type":"date","name":"Optional Date","values":["2021-01-13T15:02:00.000Z"]}]}'
 ↪  testfile.bin
aoc packages send --workspace=my_shbx_ws @json:'{"name":"Important files
 ↪  delivery","recipients":["my_shbx_name"],"metadata":{"Project
 ↪  Id":"456","Type":"Opt2","CheckThose":["Check1","Check2"],"Optional
 ↪  Date":"2021-01-13T15:02:00.000Z"}}' testfile.bin
aoc packages send --workspace=my_shbx_ws @json:'{"name":"Important files
 ↪  delivery","recipients":["my_shbx_name"]}' testfile.bin
aoc packages send @json:'{"name":"Important files delivery","recipients":["my_email_external"]}'
 ↪  --new-user-option=@json:'{"package_contact":true}' testfile.bin
aoc packages send @json:'{"name":"Important files
 ↪  delivery","recipients":["my_email_internal"],"note":"my note"}' testfile.bin
aoc packages send @json:'{"name":"Important files delivery"}' testfile.bin
 ↪  --url=my_publink_send_aoc_user --password=my_publink_send_use_pass
aoc packages send @json:'{"name":"Important files delivery"}' testfile.bin
 ↪  --url=my_publink_send_shd_inbox
aoc packages shared_inboxes list
aoc remind --username=my_user_email
aoc servers
aoc user profile modify @json:'{"name":"dummy change"}'
aoc user profile show
aoc user workspaces current
aoc user workspaces list
```

Chapter 6

Plugin: **ats**: IBM Aspera Transfer Service

ATS is usable either :

- from an AoC subscription : ascli aoc admin ats : use AoC authentication
- or from an IBM Cloud subscription : ascli ats : use IBM Cloud API key authentication

6.1 IBM Cloud ATS : creation of api key

This section is about using ATS with an IBM cloud subscription. If you are using ATS as part of AoC, then authentication is through AoC, not IBM Cloud.

First get your IBM Cloud APIkey. For instance, it can be created using the IBM Cloud web interface, or using command line:

```
ibmcloud iam api-key-create mykeyname -d 'my sample key'
```

```
OK
API key mykeyname was created

Please preserve the API key! It cannot be retrieved after it's created.

Name          mykeyname
Description   my sample key
Created At    2019-09-30T12:17+0000
API Key       my_secret_api_key_here
Locked        false
UUID          ApiKey-05b8fadf-e7fe-4bc4-93a9-6fd348c5ab1f
```

References:

- https://console.bluemix.net/docs/iam/userid_keys.html#userapikey
- https://ibm.ibmaspera.com/helpcenter/transfer-service

Then, to register the key by default for the ats plugin, create a preset. Execute:

```
ascli config preset update my_ibm_ats --ibm-api-key=my_secret_api_key_here
```

```
ascli config preset set default ats my_ibm_ats
```

```
ascli ats api_key instances
```

```
+--------------------------------------+
| instance                             |
+--------------------------------------+
| aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee |
+--------------------------------------+
```

```
ascli config preset update my_ibm_ats --instance=aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
```

```
ascli ats api_key create
```

```
+--------+-------------------------------------------+
| key    | value                                     |
+--------+-------------------------------------------+
| id     | ats_XXXXXXXXXXXXXXXXXXXXXXX               |
| secret | YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY  |
+--------+-------------------------------------------+
ascli config preset update my_ibm_ats --ats-key=ats_XXXXXXXXXXXXXXXXXXXXXXX
↪  --ats-secret=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

6.2 ATS Access key creation parameters

When creating an ATS access key, the option `params` must contain an extended value with the creation parameters. Those are directly the parameters expected by the ATS API.

6.3 Misc. Examples

Example: create access key on IBM Cloud (softlayer):

```
ascli ats access_key create --cloud=softlayer --region=ams --params=@json:'{"storage":{"type":
↪  "softlayer_swift","container":"_container_name_","credentials":{"api_key":"my_secret_here",
↪  "username":"_name_:_usr_name_"},"path":"/"},"id":"_optional_id_","name":"_optional_name_"}'
```

Example: create access key on AWS:

```
ascli ats access_key create --cloud=aws --region=eu-west-1
↪  --params=@json:'{"id":"myaccesskey","name":"laurent key
↪  AWS","storage":{"type":"aws_s3","bucket":"my-bucket","credentials":{"access_key_id":
↪  "_access_key_id_here_","secret_access_key":"my_secret_here"},"path":"/laurent"}}'
```

Example: create access key on Azure SAS:

```
ascli ats access_key create --cloud=azure --region=eastus
↪  --params=@json:'{"id":"myaccesskey","name":"laurent key
↪  azure","storage":{"type":"azure_sas","credentials":{"shared_access_signature":"https:
↪  //containername.blob.core.windows.net/blobname?sr=c&..."},"path":"/"}}'
```

(Note that the blob name is mandatory after server address and before parameters. and that parameter sr=c is mandatory.)

Example: create access key on Azure:

```
ascli ats access_key create --cloud=azure --region=eastus
↪  --params=@json:'{"id":"myaccesskey","name":"laurent key
↪  azure","storage":{"type":"azure","credentials":{"account":"myaccount","key":"myaccesskey",
↪  "storage_endpoint":"myblob"},"path":"/"}}'
```

delete all my access keys:

```
ascli ats access_key list --field=id --format=csv | ascli ats access_key delete @lines:@stdin:
↪  --bulk=yes
```

The parameters provided to ATS for access key creation are the ones of ATS API for the `POST /access_keys` endpoint.

6.4 ATS sample commands

```
ats access_key cluster ak2ibmcloud --secret=my_secret_here
ats access_key create --cloud=aws --region=my_bucket_region
↪  --params=@json:'{"id":"ak_aws","name":"my test key
↪  AWS","storage":{"type":"aws_s3","bucket":"my_bucket_name","credentials":{"access_key_id":
↪  "my_bucket_key","secret_access_key":"my_bucket_secret"},"path":"/"}}'
ats access_key create --cloud=softlayer --region=my_bucket_region
↪  --params=@json:'{"id":"ak2ibmcloud","secret":"my_secret_here","name":"my test
↪  key","storage":{"type":"ibm-s3","bucket":"my_bucket_name","credentials":{"access_key_id":
↪  "my_bucket_key","secret_access_key":"my_bucket_secret"},"path":"/"}}'
```

```
ats access_key delete ak2ibmcloud
ats access_key delete ak_aws
ats access_key entitlement ak2ibmcloud
ats access_key list --fields=name,id
ats access_key node ak2ibmcloud browse / --secret=my_secret_here
ats access_key show ak2ibmcloud
ats api_key create
ats api_key instances
ats api_key list
ats cluster clouds
ats cluster list
ats cluster show --cloud=aws --region=eu-west-1
ats cluster show 1f412ae7-869a-445c-9c05-02ad16813be2
```

Chapter 7

Plugin: `server`: IBM Aspera High Speed Transfer Server (SSH)

The `server` plugin is used for operations on Aspera HSTS using SSH authentication. It is the legacy way of accessing an Aspera Server, often used for server to server transfers. An SSH session is established, authenticated with either a password or an SSH private key, then commands `ascp` (for transfers) and `ascmd` (for file operations) are executed.

Note: The URL to be provided is usually: `ssh://_server_address_:33001`

7.1 Server sample commands

```
server browse /
server browse my_inside_folder/testfile.bin
server browse my_upload_folder/target_hot
server cp my_inside_folder/testfile.bin my_upload_folder/200KB.2
server delete my_inside_folder
server delete my_upload_folder/to.delete
server df
server download my_inside_folder/testfile.bin --to-folder=.
 ↪  --transfer-info=@json:'{"wss":false,"resume":{"iter_max":1}}'
server download my_inside_folder/testfile.bin --to-folder=my_upload_folder --transfer=node
server du /
server health transfer --to-folder=my_upload_folder --format=nagios
server info
server md5sum my_inside_folder/testfile.bin
server mkdir my_inside_folder --logger=stdout
server mkdir my_upload_folder/target_hot
server mv my_upload_folder/200KB.2 my_upload_folder/to.delete
server sync admin status --sync-info=@json:'{"name":"sync2","local":{"path":"/data/local_sync"}}'
server sync admin status --sync-info=@json:'{"name":"sync2"}'
server sync admin status mysync
 ↪  --sync-info=@json:'{"sessions":[{"name":"mysync","local_dir":"/data/local_sync"}]}'
server sync start
 ↪  --sync-info=@json:'{"instance":{"quiet":false},"sessions":[{"name":"mysync","direction":
 ↪  "pull","remote_dir":"my_inside_folder","local_dir":"/data/local_sync","reset":true}]}'
server sync start --sync-info=@json:'{"name":"sync2","local":{"path":"/data/local_sync"},
 ↪  "remote":{"path":"my_inside_folder"},"reset":true,"quiet":false}'
server upload 'TST_MED_DATAFILE' --to-folder=my_inside_folder --ts=@json:'{"multi_session":3,
 ↪  "multi_session_threshold":1,"resume_policy":"none","target_rate_kbps":100000}'
 ↪  --transfer-info=@json:'{"spawn_delay_sec":2.5,"multi_incr_udp":false}' --progressbar=yes
server upload 'faux:///test1?100m' 'faux:///test2?100m' --to-folder=/Upload
 ↪  --ts=@json:'{"target_rate_kbps":1000000,"resume_policy":"none","precalculate_job_size":true}'
server upload 'faux:///test1?100m' 'faux:///test2?100m' --to-folder=/Upload
 ↪  --ts=@json:'{"target_rate_kbps":1000000,"resume_policy":"none","precalculate_job_size":true}'
 ↪  --transfer-info=@json:'{"quiet":false}' --progress=no
```

```
server upload --sources=@ts --transfer-info=@json:'{"ascp_args":["--file-list","filelist.txt"]}'
↪  --to-folder=my_inside_folder
server upload --sources=@ts
↪  --transfer-info=@json:'{"ascp_args":["--file-pair-list","filepairlist.txt"]}'
server upload --sources=@ts --ts=@json:'{"paths":[{"source":"testfile.bin","destination":
↪  "my_inside_folder/othername4"}]}'
server upload --src-type=pair 'TST_MED_DATAFILE' my_inside_folder/othername2
↪  --notif-to=my_email_external --transfer-info=@json:'{"ascp_args":["-l","10m"]}'
server upload --src-type=pair --sources=@json:'["testfile.bin","my_inside_folder/othername3"]'
↪  --transfer-info=@json:'{"quiet":false}' --progress=no
server upload --src-type=pair testfile.bin my_upload_folder/othername5 --ts=@json:'{"cipher":
↪  "aes-192-gcm","content_protection":"encrypt","content_protection_password":"my_secret_here",
↪  "cookie":"biscuit","create_dir":true,"delete_before_transfer":false,"delete_source":false,
↪  "exclude_newer_than":1,"exclude_older_than":10000,"fasp_port":33001,"http_fallback":false,
↪  "multi_session":0,"overwrite":"diff+older","precalculate_job_size":true,
↪  "preserve_access_time":true,"preserve_creation_time":true,"rate_policy":"fair",
↪  "resume_policy":"sparse_csum","symlink_policy":"follow"}'
server upload --to-folder=my_upload_folder/target_hot --lock-port=12345
↪  --transfer-info=@json:'{"ascp_args":["--remove-after-transfer","--remove-empty-directories",
↪  "--exclude-newer-than=-8","--src-base","source_hot"]}'
↪  source_hot
```

7.2 Authentication on Server with SSH session

If SSH is the session protocol (by default i.e. not WSS), then following session authentication methods are supported:

- `password`: SSH password
- `ssh_keys`: SSH keys (Multiple SSH key paths can be provided.)

If `username` is not provided then the default transfer user `xfer` is used.

If no SSH password or key is provided and a transfer token is provided in transfer spec (option `ts`), then standard SSH bypass keys are used. Example:

```
ascli server --url=ssh://_server_address_:33001 ... --ts=@json:'{"token":"Basic _token_here_"}'
```

> Note: If you need to use the Aspera public keys, then specify an empty token: `--ts=@json:'{"token":""}'` : Aspera public SSH keys will be used, but the protocol will ignore the empty token.

The value of the `ssh_keys` option can be a single value or an `Array`. Each value is a path to a private key and is expanded (~ is replaced with the user's home folder).

Examples:

```
ascli server --ssh-keys=~/.ssh/id_rsa
ascli server --ssh-keys=@list:,~/.ssh/id_rsa
ascli server --ssh-keys=@json:'["~/.ssh/id_rsa"]'
```

For file operation command (browse, delete), the Ruby SSH client library `Net::SSH` is used and provides several options settable using option `ssh_options`.

For a list of SSH client options, refer to the Ruby documentation of Net::SSH.

Some of the 50 available SSH options:

- `verbose`
- `use_agent`
- `passphrase`

By default the SSH library will check if a local `ssh-agent` is running.

On Linux, if you get an error message such as:

```
ERROR -- net.ssh.authentication.agent: could not connect to ssh-agent: Agent not configured
```

or on Windows:

```
ERROR -- net.ssh.authentication.agent: could not connect to ssh-agent: pageant process not
↪  running
```

This means that your environment suggests to use an agent but you don't have such an SSH agent running, then:

- Check env var: `SSH_AGENT_SOCK`
- Check your file: `$HOME/.ssh/config`
- Check if the SSH key is protected with a passphrase (then, use the `passphrase` SSH option)
- Check the Ruby SSH manual
- To disable the use of `ssh-agent`, use the option `ssh_options` like this:

```
ascli server --ssh-options=@json:'{"use_agent": false}' ...
```

Note: This can also be set using a preset.

If one of the SSH private keys is passphrase-protected, then option `passphrase` can be used. It is equivalent to setting both options `ssh_options.passphrase` and `ts.ssh_private_key_passphrase`.

7.3 Other session channels for **`server`**

URL schemes `local` and `https` are also supported (mainly for testing purpose). (`--url=local:` , `--url=https://...`)

- `local` will execute `ascmd` locally, instead of using an SSH connection.
- `https` will use Web Socket Session: This requires the use of a transfer token. For example a `Basic` token can be used.

As, most of the time, SSH is used, if an `http` scheme is provided without token, the plugin will fallback to SSH and port 33001.

7.4 Examples: **`server`**

One can test the `server` application using the well known demo server:

```
ascli config initdemo
ascli server browse /aspera-test-dir-large
ascli server download /aspera-test-dir-large/200MB
```

`initdemo` creates a option preset `demoserver` and set it as default for plugin `server`.

If an SSH private key is used for authentication with a passphrase, the passphrase needs to be provided to both options: `ssh_options`, for browsing, and `ts` for transfers:

```
ascli server --url=ssh://_server_address_here_:33001 --username=_user_here_
↪  --ssh_keys=_private_key_path_here_ --passphrase=_passphrase_here_
```

Chapter 8

Plugin: **node**: IBM Aspera High Speed Transfer Server Node

This plugin gives access to capabilities provided by the HSTS node API.

The authentication is `username` and `password` or `access_key` and `secret` through options: `username` and `password`.

> Note: Capabilities of this plugin are used in other plugins which access to the node API, such as `aoc`, `ats`, `shares`.

8.1 File Operations

It is possible to do gen3/node user operations:

- browse
- transfer (upload / download / sync)
- delete
- ...

When using an access key, so called gen4/access key API is also supported through sub commands using `access_keys do self`.

Example:

- `ascli node browse /` : list files with gen3/node user API
- `ascli node access_key do self browse /` : list files with gen4/access key API

8.2 Operation **`find`** on gen4/access key

The command `find <folder> [filter_expr]` is available for gen4/access key, under `access_keys do self`.

The argument `<folder>` is mandatory and is the root from which search is performed. The argument `[filter_expr]` is optional and represent the matching criteria.

It recursively scans storage to find files/folders matching a criteria and then returns a list of matching entries.

`[filter_expr]` is either:

- Optional (default) : all files and folder are selected
- type `String` : the expression is similar to shell globbing, refer to Ruby function: `File.fnmatch`
- Type `Proc` : the expression is a Ruby lambda that takes one argument: a `Hash` that contains the current folder entry to test. Refer to the following examples.

Examples of expressions:

- Find all files and folders under `/`

```
ascli node access_keys do self find
```

- Find all text files `/Documents`

```
ascli node access_keys do self find /Documents '*.txt'
```

The following are examples of `ruby_lambda` to be provided in the following template command:

```
ascli node access_keys do self find / @ruby:'ruby_lambda'
```

Note: Single quotes are used here above to protect the whole Ruby expression from the shell. Then double quotes are used for strings in the Ruby expression to not mix with the shell.

- Find files more recent than 100 days

```
->(f){f["type"].eql?("file") and (DateTime.now-DateTime.parse(f["modified_time"]))<100}
```

- Find files older than 1 year

```
->(f){f["type"].eql?("file") and (DateTime.now-DateTime.parse(f["modified_time"]))>365}
```

- Find files larger than 1MB

```
->(f){f["type"].eql?("file") and f["size"].to_i>1000000}
```

- Filter out files beginning with `._` or named `.DS_Store`:

```
->(f){!(f["name"].start_with?("._") or f["name"].eql?(".DS_Store"))}
```

- Match files using a Ruby Regex : `\.gif$`

```
->(f){f["name"].match?(/\.gif$/)}
```

`ascli` commands can be piped in order to combine operations, such as ”find and delete”:

```
ascli node access_keys do self find / @ruby:'->(f){f["type"].eql?("file") and
↪  (DateTime.now-DateTime.parse(f["modified_time"]))>365}' --fields=path --format=csv | ascli
↪  node --bulk=yes delete @lines:@stdin:
```

Note: the pipe | character on the last line.

8.3 Central

The central subcommand uses the reliable query API (session and file). It allows listing transfer sessions and transferred files.

Filtering can be applied:

```
ascli node central file list
```

By providing the `validator` option, offline transfer validation can be done.

Note: This API will probably be deprecated in the future.

8.4 Sync

There are three sync types of operations:

- `sync`: perform a local sync, by executing `async` locally
- `async`: calls legacy async API on node : `/async`
- `ssync` : calls newer async API on node : `/asyncs`

8.5 FASP Stream

It is possible to start a FASPStream session using the node API:

Use the command `ascli node stream create --ts=@json:<value>`, with transfer-spec:

```
{"direction":"send","source":"udp://233.3.3.4:3000?loopback=1&ttl=2","destination":"udp://233.3.
↪ 3.3:3001/","remote_host":"localhost","remote_user":"stream","remote_password":"my_pass_here"}
```

8.6 Watchfolder

Refer to Aspera documentation for watch folder creation.

`ascli` supports remote operations through the node API. Operations are:

- Start watchd and watchfolderd services running as a system user having access to files
- configure a watchfolder to define automated transfers

```
ascli node service create @json:'{"id":"mywatchd","type":"WATCHD","run_as":{"user":"user1"}}'
ascli node service create
↪ @json:'{"id":"mywatchfolderd","type":"WATCHFOLDERD","run_as":{"user":"user1"}}'
ascli node watch_folder create @json:'{"id":"mywfolder","source_dir":"/watch1","target_dir":"/",
↪ "transport":{"host":"10.25.0.4","user":"user1","pass":"mypassword"}}'
```

8.7 Out of Transfer File Validation

Follow the Aspera Transfer Server configuration to activate this feature.

```
ascli node central file list --validator=ascli
↪ --data=@json:'{"file_transfer_filter":{"max_result":1}}'
```

```
+--------------+--------------+------------+--------------------------------------+
| session_uuid |    file_id   |   status   |              path                    |
+--------------+--------------+------------+--------------------------------------+
| 1a74444c-... | 084fb181-... | validating | /home/xfer.../PKG - my title/200KB.1 |
+--------------+--------------+------------+--------------------------------------+
```

```
ascli node central file update --validator=ascli --data=@json:'{"files":[{"session_uuid":
↪ "1a74444c-...","file_id": "084fb181-...","status": "completed"}]}'
```

```
updated
```

8.8 Example: SHOD to ATS

Scenario: Access to a Shares on Demand (SHOD) server on AWS is provided by a partner. We need to transfer files from this third party SHOD instance into our Azure BLOB storage. Simply create an Aspera Transfer Service instance, which provides access to the node API. Then create a configuration for the SHOD instance in the configuration file: in section ”shares”, a configuration named: aws_shod. Create another configuration for the Azure ATS instance: in section ”node”, named azure_ats. Then execute the following command:

```
ascli node download /share/sourcefile --to-folder=/destination_folder --preset=aws_shod
↪ --transfer=node --transfer-info=@preset:azure_ats
```

This will get transfer information from the SHOD instance and tell the Azure ATS instance to download files.

8.9 node file information

When node api is used with an Access key, extra information can be retrieved, such as preview.

Note: Display of preview on terminal requires installation of extra gem: `rmagick`

```
dnf install -y ImageMagick-devel
gem install rmagick rainbow
```

For example, it is possible to display the preview of a file, if it exists, using an access key on node:

```
ascli node access_key do self thumbnail /preview_samples/Aspera.mpg
```

Previews are mainly used in AoC, this also works with AoC:

```
ascli aoc files thumbnail /preview_samples/Aspera.mpg
```

Note: To specify the file by its file id, use the selector syntax: `%id:_file_id_here_`

Note: To force textual display of the preview on iTerm, prefix command with: `env -u TERM_PROGRAM -u LC_TERMINAL`

8.10 Create access key

```
ascli node access_key create @json:'{"id":"myaccesskey","secret":"my_secret_here","storage":
↪ {"type":"local","path":"/data/mydir"}}'
```

Note: The `id` and `secret` are optional. If not provided, they will be generated and returned in the result.

Access keys support extra overriding parameters using parameter: `configuration` and sub keys `transfer` and `server`. For example, an access key can be modified or created with the following options:

```
{"configuration":{"transfer":{"target_rate_cap_kbps":500000}}}
```

The list of supported options can be displayed using command:

```
ascli node info --field=@ruby:'/^access_key_configuration_capabilities.*/'
```

8.11 Generate and use bearer token

Bearer tokens are part of the gen4/access key API. It follows the model of OAuth 2. For example they are used in Aspera on Cloud. This is also available for developers for any application integrating Aspera. In this API, files, users and groups are identified by an id (a String, e.g. `"125"`, not necessarily numerical).

Bearer tokens are typically generated by the authentification application, and then recognized by the node API. A bearer token is authorized on the node by creating `permissions` on a folder.

Bearer tokens can be generated using command `bearer_token`: it takes two arguments:

- the private key used to sign the token
- the token information, which is a `Hash` containing the following elements:

parameter	Default	type	description
user_id	-	Mandatory	Identifier of user
scope	`node.<access_key>:<_scope>`	Mandatory	API scope, e.g. `node.<access_key>:<node scope>`
expires_at	`now+<_validity>`	Mandatory	Format: `%Y-%m-%dT%H:%M:%SZ` e.g. `2021-12-31T23:59:59Z`
auth_type	`access_key`	Optional	`access_key`, `node_user`
group_ids	-	Optional	List of group ids
organization_id	-	Optional	Organization id
watermarking_json_base64	-	Optional	Watermarking information (not used)
_scope	`user:all`	Special	Either `user:all` or `admin:all`
_validity	86400	Special	Validity in seconds from now.

Note: For convenience, `ascli` provides additional parameters `_scope` and `_validity`. They are not part of the API and are removed from the final payload. They are used respectively to build the default value of `scope` and `expires_at`.

8.11.1 Bearer token: environment

- If a self-managed Aspera node is used, then a ”node user admin” must be created: It has no `docroot` but has at least one file restriction (for testing, one can use `*` to accept creation of an access key with any storage root path). Refer to the Aspera HSTS documentation.

- If Cloud Pak for integration is used, then the node admin is created automatically.
- If Aspera on Cloud or ATS is used, then the SaaS API for access key creation is used.
- An access key shall be created to grant access for transfers to its storage. The access_key and its secrets represent an administrative access to the storage as it has access rights to the whole storage of the access key.

8.11.2 Bearer token: preparation

Let's assume that the access key was created, and a default configuration is set to use this access key.

- Create a private key (organization key) that will be used to sign bearer tokens:

```
my_private_pem=./myorgkey.pem
ascli conf genkey $my_private_pem
```

  Note: This key is not used for authentication, it is used to sign bearer tokens. Refer to section private key for more details on generation.

- The corresponding public key shall be placed as an attribute of the access key:

```
ascli node access_key set_bearer_key self @file:$my_private_pem
```

  Note: Either the public or private key can be provided, and only the public key is used. This will enable to check the signature of the bearer token.

  Alternatively, use the following equivalent command, as `ascli` kindly extracts the public key with extension `.pub`:

```
ascli node access_key modify %id:self @ruby:'{token_verification_key:
↪  File.read("'$my_private_pem'.pub")}'
```

8.11.3 Bearer token: configuration for user

- Select a folder for which we want to grant access to a user, and get its identifier:

```
my_folder_id=$(ascli node access_key do self show / --fields=id)
```

  Note: Here we simply select `/`, but any folder can be selected in the access key storage.

- Let's designate a user by its id:

```
my_user_id=777
```

  Note: This is an arbitrary identifier, typically managed by the web application. Not related to Linux user ids or anything else.

- Grant this user access to the selected folder:

```
ascli node access_key do self permission %id:$my_folder_id create
↪  @json:'{"access_type":"user","access_id":"'$my_user_id'"}'
```

- Create a Bearer token for the user:

```
ascli node bearer_token @file:./myorgkey.pem
↪  @json:'{"user_id":"'$my_user_id'","_validity":3600}' > bearer.txt
```

8.11.4 Bearer token: user side

Now, let's assume we are the user, the only information received are:

- the url of the node API
- a Bearer token
- a file `id` for which we have access

Let's use it:

```
ascli node -N --url=... --password="Bearer $(cat bearer.txt)" --root-id=$my_folder_id access_key
↪  do self br /
```

8.12 Node sample commands

```
node --url=https://tst.example.com/path --password="Bearer $bearer_666" --root-id=$root_id
↪  access_key do self br /
node access_key create @json:'{"id":"my_username","secret":"my_password_here","storage":{"type":
↪  "local","path":"/"}}'
node access_key delete my_username
node access_key do my_ak_name browse /
node access_key do my_ak_name delete /folder2
node access_key do my_ak_name delete testfile1
node access_key do my_ak_name download testfile1 --to-folder=.
node access_key do my_ak_name find my_test_folder
node access_key do my_ak_name find my_test_folder @ruby:'->(f){f["name"].end_with?(".jpg")}'
node access_key do my_ak_name find my_test_folder exec:'f["name"].end_with?(".jpg")'
node access_key do my_ak_name mkdir /folder1
node access_key do my_ak_name node_info /
node access_key do my_ak_name rename /folder1 folder2
node access_key do my_ak_name show %id:1
node access_key do my_ak_name show /testfile1
node access_key do my_ak_name upload 'faux:///testfile1?1k' --default_ports=no
node access_key do self permission %id:$root_id create
↪  @json:'{"access_type":"user","access_id":"666"}'
node access_key do self show / --fields=id
node access_key list
node access_key set_bearer_key self @file:my_private_key
node api_details
node async bandwidth 1
node async counters 1
node async files 1
node async list
node async show 1
node async show ALL
node basic_token
node bearer_token @file:my_private_key @json:'{"user_id":"666"}'
node browse / -r
node delete @list:,my_upload_folder/todelete,my_upload_folder/tdlink,my_upload_folder/delfile
node delete my_upload_folder/testfile.bin
node download my_upload_folder/testfile.bin --to-folder=.
node health
node info --fpac='function FindProxyForURL(url,host){return "DIRECT"}'
node license
node mkdir my_upload_folder/todelete
node mkfile my_upload_folder/delfile1 "hello world"
node mklink my_upload_folder/todelete my_upload_folder/tdlink
node rename my_upload_folder delfile1 delfile
node search / --query=@json:'{"sort":"mtime"}'
node service create @json:'{"id":"service1","type":"WATCHD","run_as":{"user":"user1"}}'
node service delete service1
node service list
node space /
node ssync bandwidth %name:my_node_sync
node ssync counters %name:my_node_sync
node ssync create @json:'{"configuration":{"name":"my_node_sync","local":{"path":
↪  "my_local_path"},"remote":{"host":"my_host","port":my_port,"user":"my_username","pass":
↪  "my_password_here","path":"my_remote_path"}}}'
node ssync delete %name:my_node_sync
node ssync files %name:my_node_sync
node ssync list
node ssync show %name:my_node_sync
node ssync start %name:my_node_sync
node ssync state %name:my_node_sync
node ssync stop %name:my_node_sync
node ssync summary %name:my_node_sync
```

```
node sync admin status --sync-info=@json:'{"name":"my_node_sync2","reset":true,"direction":
↪  "pull","local":{"path":"/data/local_sync"},"remote":{"path":"/aspera-test-dir-tiny"}}'
node sync admin status --sync-info=@json:'{"sessions":[{"name":"my_node_sync1","direction":
↪  "pull","local_dir":"/data/local_sync","remote_dir":"/aspera-test-dir-tiny","reset":true}]}'
node sync start --sync-info=@json:'{"name":"my_node_sync2","reset":true,"direction":"pull",
↪  "local":{"path":"/data/local_sync"},"remote":{"path":"/aspera-test-dir-tiny"}}'
node sync start --sync-info=@json:'{"sessions":[{"name":"my_node_sync1","direction":"pull",
↪  "local_dir":"/data/local_sync","remote_dir":"/aspera-test-dir-tiny","reset":true}]}'
node transfer list --query=@json:'{"active_only":true}'
node transfer sessions
node upload --to-folder=my_upload_folder --sources=@ts --ts=@json:'{"paths":[{"source":"/aspera-
↪  test-dir-small/10MB.2"}],"precalculate_job_size":true}' --transfer=node
↪  --transfer-info=@json:'{"url":"https://node.example.com/path@","username":"my_username",
↪  "password":"my_password_here"}'
node upload --username=my_ak_name --password=my_ak_secret testfile.bin
node upload testfile.bin --to-folder=my_upload_folder --ts=@json:'{"target_rate_cap_kbps":10000}'
```

Chapter 9

Plugin: **faspex5**: IBM Aspera Faspex v5

IBM Aspera's newer self-managed application.

3 authentication methods are supported:

- jwt : general purpose, private-key based authentication
- link : public link authentication
- web : requires authentication with web browser
- boot : use authentication token copied from browser (experimental)

For a quick start, one can use the wizard, which will help creating a option preset:

```
ascli config wizard
```

```
argument: url> faspex5.example.com
Multiple applications detected:
+---------+-----------------------------------------+-------------+
| product | url                                     | version     |
+---------+-----------------------------------------+-------------+
| faspex5 | https://faspex5.example.com/aspera/faspex | F5.0.6      |
| server  | ssh://faspex5.example.com:22              | OpenSSH_8.3 |
+---------+-----------------------------------------+-------------+
product> faspex5
Using: Faspex at https://faspex5.example.com/aspera/faspex
Please provide the path to your private RSA key, or nothing to generate one:
option: pkeypath>
Using existing key:
/Users/someuser/.aspera/ascli/my_key
option: username> someuser@example.com
Ask the ascli client id and secret to your Administrator.
Admin should login to: https://faspex5.example.com/aspera/faspex
Navigate to: ::  → Admin → Configurations → API clients
Create an API client with:
- name: ascli
- JWT: enabled
Then, logged in as someuser@example.com go to your profile:
() → Account Settings → Preferences -> Public Key in PEM:
-----BEGIN PUBLIC KEY-----
redacted
-----END PUBLIC KEY-----
Once set, fill in the parameters:
option: client_id> _my_key_here_
option: client_secret> ****
Preparing preset: faspex5_example_com_user
Setting config preset as default for faspex5
Done.
You can test with:
ascli faspex5 user profile show
Saving config file.
```

Note: Include the public key `BEGIN` and `END` lines when pasting in the user profile.

For details on the JWT method, see the following section.

9.1 Faspex 5 JWT authentication

This is the general purpose and recommended method to use.

Activation is in two steps:

- The admninistrator must create an API client in Faspex with JWT support

  This operation is generally done only once:

  - As Admin, Navigate to the web UI: Admin → Configurations → API Clients → Create
  - Give a name, like `ascli`
  - Activate JWT
  - There is an option to set a general public key allowing the owner of the private key to impersonate any user. Unless you want to do this, leave this field empty.
  - Click on `Create` Button
  - Take note of Client Id (and Client Secret, but not used in current version)

- The user uses a private key and sets the public key in his faspex 5 profile

  This operation is done by each user using the CLI.

  - As user, click on the user logo, left to the app switcher on top right.
  - Select `Account Settings`
  - on the bottom in the text field: `Public key in PEM format` paste the public key corresponding to the private key used by the user.

  Note: If you don't have any refer to section Private Key

Then use these options:

```
--auth=jwt
--client-id=_client_id_here_
--client-secret=my_secret_here
--username=_username_here_
--private-key=@file:.../path/to/key.pem
```

Note: The `private_key` option must contain the PEM value (not file path) of the private key which can be read from a file using the modifier: `@file:`, e.g. `@file:/path/to/key.pem`.

As usual, typically a user will create preset to avoid having to type these options each time.

Example:

```
ascli conf preset update myf5 --auth=jwt --client-id=_client_id_here_
 ↪  --client-secret=my_secret_here --username=_username_here_
 ↪  --private-key=@file:.../path/to/key.pem

ascli conf preset set default faspx5 myf5

ascli faspex5 user profile show
```

9.2 Faspex 5 web authentication

The admninistrator must create an API client in Faspex for an external web app support:

- As Admin, Navigate to the web UI: Admin → Configurations → API Clients → Create
- Do not Activate JWT
- Set Redirect URI to `https://127.0.0.1:8888`
- Click on `Create` Button
- Take note of Client Id (and Client Secret, but not used in current version)

The user will use the following options:

```
--auth=web
--client-id=_client_id_here_
--client-secret=my_secret_here
--redirect-uri=https://127.0.0.1:8888
```

9.3 Faspex 5 bootstrap authentication

For `boot` method: (will be removed in future)

- As user: Open a Web Browser
- Start developer mode
- Login to Faspex 5
- Find the first API call with `Authorization` header, and copy the value of the token (series of base64 values with dots)

Use this token as password and use `--auth=boot`.

```
ascli conf preset update f5boot --url=https://localhost/aspera/faspex --auth=boot
↪  --password=_token_here_
```

9.4 Faspex 5 sample commands

Most commands are directly REST API calls. Parameters to commands are carried through option `query`, as extended value, for `list`, or through positional argument for creation. One can conveniently use the JSON format with prefix `@json:`.

> Note: The API is listed in Faspex 5 API Reference under IBM Aspera Faspex API.

```
faspex5 admin res accounts list
faspex5 admin res contacts list
faspex5 admin res jobs list
faspex5 admin res metadata_profiles list
faspex5 admin res node list
faspex5 admin res oauth_clients list
faspex5 admin res registrations list
faspex5 admin res saml_configs list
faspex5 admin res shared_inboxes invite %name:'ascli shinbox' johnny@example.com
faspex5 admin res shared_inboxes list
faspex5 admin res shared_inboxes members %name:'ascli shinbox' create %name:john@example.com
faspex5 admin res shared_inboxes members %name:'ascli shinbox' delete %name:john@example.com
faspex5 admin res shared_inboxes members %name:'ascli shinbox' delete %name:johnny@example.com
faspex5 admin res shared_inboxes members %name:'ascli shinbox' list
faspex5 admin res workgroups list
faspex5 admin smtp show
faspex5 admin smtp test my_email_external
faspex5 bearer_token
faspex5 gateway https://localhost:12345/aspera/faspex
faspex5 health
faspex5 packages list --box=my_shinbox
faspex5 packages list --box=my_workgroup --group-type=workgroups
faspex5 packages list --query=@json:'{"mailbox":"inbox","state":["released"]}'
faspex5 packages receive "my_package_id" --to-folder=.
↪  --ts=@json:'{"content_protection_password":"abc123_yo"}'
faspex5 packages receive --box=my_shinbox "my_package_id" --to-folder=.
faspex5 packages receive --box=my_workgroup --group-type=workgroups "my_package_id" --to-folder=.
faspex5 packages receive ALL --once-only=yes --to-folder=.
faspex5 packages receive INIT --once-only=yes
faspex5 packages send @json:'{"title":"test
↪  title","recipients":["my_shinbox"],"metadata":{"Options":"Opt1","TextInput":"example text"}}'
↪  testfile.bin
faspex5 packages send @json:'{"title":"test title","recipients":["my_workgroup"]}' testfile.bin
```

```
faspex5 packages send @json:'{"title":"test title","recipients":[{"name":"my_username"}]my_meta}'
↪  testfile.bin --ts=@json:'{"content_protection_password":"my_passphrase_here"}'
faspex5 packages show "my_package_id"
faspex5 packages show --box=my_shinbox "my_package_id"
faspex5 packages show --box=my_workgroup --group-type=workgroups "my_package_id"
faspex5 packages status "my_package_id"
faspex5 postprocessing
↪  @json:'{"url":"https://localhost:8443/domain","processing":{"script_folder":"tests"},
↪  "certificate":{"key":"../local/k","cert":"../local/c","chain":"../local/ch"}}'
faspex5 shared browse %name:my_src
faspex5 shared list
faspex5 user profile modify @json:'{"preference":{"connect_disabled":false}}'
faspex5 user profile show
```

9.5 Faspex 5: inbox selection

By default, package operations (send, receive, list) are done on the user's inbox.

To select another inbox, use option `box` with one of the following values:

- inbox
- inbox_history
- inbox_all
- inbox_all_history
- outbox
- outbox_history
- pending
- pending_history
- all
- ALL (only admin)
- name of a shared inbox or workgroup

  Note: specify if the box is a shared inbox or a workgroup using option `group_type` with either `shared_inboxes` or `workgroups`

9.6 Faspex 5: Send a package

The `Hash` creation parameter provided to command `faspex5 packages send` corresponds to the Faspex 5 API: `POST /packages`.

Required fields are `title` and `recipients`. Example using `@json:` format:

```
{"title":"some title","recipients":[{"recipient_type":"user","name":"user@example.com"}]}
```

`recipient_type` is one of (Refer to API):

- user
- workgroup
- external_user
- distribution_list
- shared_inbox

`ascli` adds some convenience: The API expects the field `recipients` to be an `Array` of `Hash`, each with field `name` and optionally `recipient_type`. It is also possible to provide an `Array` of `String`, with simply a recipient name. Then `ascli` will lookup existing contacts among all possible types, use it if a single match is found, and set the `name` and `recipient_type` accordingly. Else an exception is sent.

Note: The lookup is case insensitive and on partial matches.

```
{"title":"some title","recipients":["user@example.com"]}
```

If the lookup needs to be only on certain types, you can specify the field: `recipient_types` with either a single value or an Array of values (from the list above). e.g. :

```
{"title":"test
↪  title","recipient_types":"user","recipients":["user1@example.com","user2@example.com"]}
```

9.7 Faspex 5: Send a package with metadata

The interface is the one of the API (Refer to API documentation, or look at request in browser):

```
ascli faspex5 packages send @json:'{"title":"test title","recipients":["my shared
↪  inbox"],"metadata":{"Confidential":"Yes","Drop menu":"Option 1"}}' 'faux:///test1?k1'
```

Basically, add the field `metadata`, with one key per metadata and the value is directly the metadata value.

9.8 Faspex 5: Receive a package

The (numeric) identifier of the package t receive is given as argument to command `faspex5 packages receive`.

Note: option `box` applies.

9.9 Faspex 5: List packages

The following parameters in option `query` are supported:

- `q` : a filter on name (case insensitive, matches if value is contained in name)
- `max` : maximum number of items to retrieve (stop pages when the maximum is passed)
- `pmax` : maximum number of pages to request (stop pages when the maximum is passed)
- `offset` : native api parameter, in general do not use (added by `ascli`)
- `limit` : native api parameter, number of items par api call, in general do not use (added by `ascli`)

Admin only: If the value `ALL` is provided to option `box`, then all packages are selected.

9.10 Faspex 5: List all shared inboxes and workgroups

If ypou are a regular, user, list workgroups you belong to:

```
ascli faspex5 admin res workgroup list
```

If you are admin or manager, add option: `--query=@json:'{"all":true}'`, this will list items you manage, even if you do not belong to them.

```
ascli faspex5 admin res shared list --query=@json:'{"all":true}' --fields=id,name
```

Shared inbox members can also be listed, added, removed, and external users can be invited to a shared inbox.

```
ascli faspex5 admin res shared_inboxes invite '%name:the shared inbox' john@example.com
```

It is equivalent to:

```
ascli faspex5 admin res shared_inboxes invite '%name:the shared inbox'
↪  @json:'{"email_address":"john@example.com"}'
```

Other payload parameters are possible the `Hash` value:

```
{"description":"blah","prevent_http_upload":true,"custom_link_expiration_policy":false,
↪  "invitation_expires_after_upload":false,"set_invitation_link_expiration":false,
↪  "invitation_expiration_days":3
```

9.11 Faspex 5: Create Metadata profile

```
ascli faspex5 admin res metadata_profiles create @json:'{"name":"the
↪  profile","default":false,"title":{"max_length":200,"illegal_chars":[]},"note":{"max_length":
↪  400,"illegal_chars":[],"enabled":false},"fields":[{"ordering":0,"name":"field1","type":
↪  "text_area","require":true,"illegal_chars":[],"max_length":100},{"ordering":1,"name":"fff2",
↪  "type":"option_list","require":false,"choices":["opt1","opt2"]}]}'
```

9.12 Faspex 5: Create a Shared inbox with specific metadata profile

```
ascli faspex5 admin res shared create @json:'{"name":"the shared inbox","metadata_profile_id":1}'
```

9.13 Faspex 5: List content in Shared folder and send package from remote source

```
ascli faspex5 shared_folders list
```

```
+----+----------+---------+-----+
| id | name     | node_id | ... |
+----+----------+---------+-----+
| 3  | partages | 2       | ... |
+----+----------+---------+-----+
```

```
ascli faspex5 shared_folders br %name:partages /folder
```

```
ascli faspex5 packages send @json:'{"title":"hello","recipients":[{"name":"_recipient_here_"}]}'
 ↪  --shared-folder=%name:partages /folder/file
```

Note: The shared folder can be identified by its numerical `id` or by name using percent selector: `%<field>:<value>`. e.g. `--shared-folder=3`

9.14 Faspex 5: receive all packages (cargo)

To receive all packages, only once, through persistency of already received packages:

```
ascli faspex5 packages receive ALL --once-only=yes
```

To initialize, and skip all current package so that next time `ALL` is used, only newer packages are downloaded:

```
ascli faspex5 packages receive INIT --once-only=yes
```

9.15 Faspex 5: Faspex 4-style postprocessing

`ascli` provides command `postprocessing` in plugin `faspex5` to emulate Faspex 4 postprocessing. It implements Faspex 5 web hooks, and calls a local script with the same environment as Faspex 4.

It is invoked like this:

```
ascli faspex5 postprocessing @json:'{"url":"http://localhost:8080/processing"}'
```

The following parameters are supported:

parameter	type	default	description
url	string	http://localhost:8080	Defines the base url on which requests are listened
certificate	hash	nil	used to define certificate if https is used
certificate.key	string	nil	path to private key file
certificate.cert	string	nil	path to certificate
certificate.chain	string	nil	path to intermediary certificates
processing	hash	nil	behavior of post processing
processing.script_folder	string	.	prefix added to script path
processing.fail_on_error	bool	false	if true and process exit with non zero, then fail
processing.timeout_seconds	integer	60	processing script is killed if takes more time

Parameter `url` defines:

- if http or https is used
- the local port
- the ”domain”, i.e. main path of url

When a request is received the following happens:

- the processor get the path of the url called
- it removes the ”domain
- it prepends it with the value of `script_folder`
- it executes the script
- upon success, a success code is returned

In Faspex 5, configure like this:

`Webhook endpoint URI:http://localhost:8080/processing/script1.sh`

Then, the postprocessing script executed will be `script1.sh`.

Environment variables at set to the values provided by the web hook which are the same as Faspex 4 postprocessing.

Chapter 10

Plugin: **faspex**: IBM Aspera Faspex v4

Notes:

- The command `v4` requires the use of APIv4, refer to the Faspex Admin manual on how to activate.
- For full details on Faspex API, refer to: Reference on Developer Site

10.1 Listing Packages

Command: `faspex package list`

10.1.1 Option **box**

By default it looks in box `inbox`, but the following boxes are also supported: `archive` and `sent`, selected with option `box`.

10.1.2 Option **recipient**

A user can receive a package because the recipient is:

- the user himself (default)
- the user is member of a dropbox/workgroup: filter using option `recipient` set with value `*<name of dropbox/workgroup>`

10.1.3 Option **query**

As inboxes may be large, it is possible to use the following query parameters:

- `count` : (native) number items in one API call (default=0, equivalent to 10)
- `page` : (native) id of page in call (default=0)
- `startIndex` : (native) index of item to start, default=0, oldest index=0
- `max` : maximum number of items
- `pmax` : maximum number of pages

(SQL query is `LIMIT <startIndex>, <count>`)

The API is listed in Faspex 4 API Reference under "Services (API v.3)".

If no parameter `max` or `pmax` is provided, then all packages will be listed in the inbox, which result in paged API calls (using parameter: `count` and `page`). By default `count` is `0` (10), it can be increased to issue less HTTP calls.

10.1.4 Example: list packages in dropbox

```
ascli faspex package list --box=inbox --recipient='*my_dropbox'
↪  --query=@json:'{"max":20,"pmax":2,"count":20}'
```

List a maximum of 20 items grouped by pages of 20, with maximum 2 pages in received box (inbox) when received in dropbox *my_dropbox.

10.2 Receiving a Package

The command is `package recv`, possible methods are:

- provide a package id with option `id`
- provide a public link with option `link`
- provide a `faspe:` URI with option `link`

```
ascli faspex package recv 12345
ascli faspex package recv --link=faspe://...
```

If the package is in a specific dropbox/workgroup, add option `recipient` for both the `list` and `recv` commands.

```
ascli faspex package list --recipient='*dropbox_name'
ascli faspex package recv 125 --recipient='*dropbox_name'
```

if `id` is set to `ALL`, then all packages are downloaded, and if option `once_only`is used, then a persistency file is created to keep track of already downloaded packages.

10.3 Sending a Package

The command is `faspex package send`. Package information (title, note, metadata, options) is provided in option `delivery_info`. The contents of `delivery_info` is directly the contents of the `send` v3 API of Faspex 4.

Example:

```
ascli faspex package send --delivery-info=@json:'{"title":"my
↪ title","recipients":["someuser@example.com"]}' --url=https://faspex.corp.com/aspera/faspex
↪ --username=foo --password=bar /tmp/file1 /home/bar/file2
```

If the recipient is a dropbox or workgroup: provide the name of the dropbox or workgroup preceded with * in the `recipients` field of the `delivery_info` option: `"recipients":["*MyDropboxName"]`

Additional optional parameters in `delivery_info`:

- Package Note: : `"note":"note this and that"`
- Package Metadata: `"metadata":{"Meta1":"Val1","Meta2":"Val2"}`

It is possible to send from a remote source using option `remote_source`, providing either the numerical id, or the name of the remote source using percent selector: `%name:<name>`.

Remote source can be browsed if option `storage` is provided. `storage` is a `Hash` extended value. The key is the storage name, as listed in `source list` command. The value is a `Hash` with the following keys:

- `node` is a `Hash` with keys: `url`, `username`, `password`
- `path` is the subpath inside the node, as configured in Faspex

10.4 Email notification on transfer

Like for any transfer, a notification can be sent by email using parameters: `notif_to` and `notif_template` .

Example:

```
ascli faspex package send --delivery-info=@json:'{"title":"test pkg
↪ 1","recipients":["aspera.user1@gmail.com"]}' ~/Documents/Samples/200KB.1
↪ --notif-to=aspera.user1@gmail.com --notif-template=@ruby:'%Q{From: <%=from_name%>
↪ <<%=from_email%>>\nTo: <<%=to%>>\nSubject: Package sent:
↪ <%=ts["tags"]["aspera"]["faspex"]["metadata"]["_pkg_name"]%> files received\n\nTo user:
↪ <%=ts["tags"]["aspera"]["faspex"]["recipients"].first["email"]%>}'
```

In this example the notification template is directly provided on command line. Package information placed in the message are directly taken from the tags in transfer spec. The template can be placed in a file using modifier: `@file:`

10.5 Operation on dropboxes

Example:

```
ascli faspex v4 dropbox create @json:'{"dropbox":{"e_wg_name":"test1","e_wg_desc":"test1"}}'
ascli faspex v4 dropbox list
ascli faspex v4 dropbox delete 36
```

10.6 Remote sources

Faspex lacks an API to list the contents of a remote source (available in web UI). To workaround this, the node API is used, for this it is required to add a section ":storage" that links a storage name to a node config and sub path.

Example:

```
my_faspex_conf:
  url: https://10.25.0.3/aspera/faspex
  username: admin
  password: MyUserPassword
  storage:
    my_storage:
      node: "@preset:my_faspex_node"
      path: /mydir
my_faspex_node:
  url: https://10.25.0.3:9092
  username: node_faspex
  password: MyNodePassword
```

In this example, a faspex storage named `my_storage` exists in Faspex, and is located under the docroot in `/mydir` (this must be the same as configured in Faspex). The node configuration name is `my_faspex_node` here.

Note: the v4 API provides an API for nodes and shares.

10.7 Automated package download (cargo)

It is possible to tell `ascli` to download newly received packages, much like the official cargo client, or drive. Refer to the same section in the Aspera on Cloud plugin:

```
ascli faspex packages recv ALL --once-only=yes --lock-port=12345
```

10.8 Faspex 4 sample commands

```
faspex address_book
faspex dropbox list --recipient="*my_dbx"
faspex health
faspex login_methods
faspex me
faspex package list --box=sent --fields=package_id --format=csv --display=data
 ↪  --query=@json:'{"max":1}'
faspex package list --fields=package_id --format=csv --display=data --query=@json:'{"max":1}'
faspex package list --query=@json:'{"max":5}'
faspex package list --recipient="*my_dbx" --format=csv --fields=package_id
 ↪  --query=@json:'{"max":1}'
faspex package list --recipient="*my_wkg" --format=csv --fields=package_id
 ↪  --query=@json:'{"max":1}'
faspex package recv "my_package_id" --to-folder=.
faspex package recv "my_package_id" --to-folder=. --box=sent
faspex package recv --to-folder=. --link=https://app.example.com/recv_from_user_path
faspex package recv ALL --to-folder=. --once-only=yes --query=@json:'{"max":10}'
faspex package recv my_pkgid --recipient="*my_dbx" --to-folder=.
faspex package recv my_pkgid --recipient="*my_wkg" --to-folder=.
```

```
faspex package send --delivery-info=@json:'{"title":"Important files
↪  delivery","recipients":["*my_dbx"]}' testfile.bin
faspex package send --delivery-info=@json:'{"title":"Important files
↪  delivery","recipients":["*my_wkg"]}' testfile.bin
faspex package send --delivery-info=@json:'{"title":"Important files
↪  delivery","recipients":["my_email_internal","my_username"]}' testfile.bin
faspex package send --delivery-info=@json:'{"title":"TIMESTAMP package
↪  remote","recipients":["my_email_internal"]}' --remote_source=%name:my_src sample_source.txt
faspex package send --link=https://app.example.com/send_to_dropbox_path
↪  --delivery-info=@json:'{"title":"Important files delivery"}' testfile.bin
faspex package send --link=https://app.example.com/send_to_user_path
↪  --delivery-info=@json:'{"title":"Important files delivery"}' testfile.bin
faspex source info %name:my_src --storage=@preset:faspex4_storage
faspex source list
faspex source node %name:my_src br / --storage=@preset:faspex4_storage
faspex v4 dmembership list
faspex v4 dropbox list
faspex v4 metadata_profile list
faspex v4 user list
faspex v4 wmembership list
faspex v4 workgroup list
```

Chapter 11

Plugin: **shares**: IBM Aspera Shares v1

Aspera Shares supports the ”node API” for the file transfer part.

11.1 Shares 1 sample commands

```
shares admin group list
shares admin node list
shares admin share list --fields=DEF,-status,status_message
shares admin share user_permissions 1 list
shares admin user add --type=ldap the_name
shares admin user app_authorizations 1 modify @json:'{"app_login":true}'
shares admin user app_authorizations 1 show
shares admin user import --type=saml @json:'{"id":"the_id","name_id":"the_name"}'
shares admin user list
shares admin user list --type=local
shares admin user share_permissions 1 list
shares admin user share_permissions 1 show 1
shares files browse /
shares files delete my_share1/testfile.bin
shares files download --to-folder=. my_share1/testfile.bin
shares files download --to-folder=. my_share1/testfile.bin --transfer=httpgw
↪  --transfer-info=@json:'{"url":"https://my_http_gw_fqdn_port/aspera/http-gwy"}'
shares files upload --to-folder=my_share1 'faux:///testfile?1m' --transfer=httpgw
↪  --transfer-info=@json:'{"url":"https://my_http_gw_fqdn_port/aspera/http-gwy","synchronous":
↪  true,"api_version":"v1","upload_chunk_size":100000}'
shares files upload --to-folder=my_share1 testfile.bin
shares files upload --to-folder=my_share1 testfile.bin --transfer=httpgw
↪  --transfer-info=@json:'{"url":"https://my_http_gw_fqdn_port/aspera/http-gwy"}'
shares health
```

Chapter 12

Plugin: **console**: IBM Aspera Console

12.1 Console sample commands

```
console health
console transfer current list
console transfer smart list
console transfer smart sub my_smart_id
 ↪  @json:'{"source":{"paths":["my_smart_file"]},"source_type":"user_selected"}'
```

Chapter 13

Plugin: **orchestrator**:IBM Aspera Orchestrator

13.1 Orchestrator sample commands

```
orchestrator health
orchestrator info
orchestrator plugins
orchestrator processes
orchestrator workflow details my_workflow_id
orchestrator workflow export my_workflow_id
orchestrator workflow inputs my_workflow_id
orchestrator workflow list
orchestrator workflow start my_workflow_id @json:'{"Param":"world !"}'
orchestrator workflow start my_workflow_id @json:'{"Param":"world !"}'
 ↪  --result=ResultStep:Complete_status_message
orchestrator workflow status ALL
orchestrator workflow status my_workflow_id
```

Chapter 14

Plugin: **cos**: IBM Cloud Object Storage

The IBM Cloud Object Storage provides the possibility to execute transfers using FASP. It uses the same transfer service as Aspera on Cloud, called Aspera Transfer Service (ATS). Available ATS regions: https://status.aspera.io

There are two possibilities to provide credentials. If you already have the endpoint, apikey and CRN, use the first method. If you don't have credentials but have access to the IBM Cloud console, then use the second method.

14.1 Using endpoint, apikey and Resource Instance ID (CRN)

If you have those parameters already, then following options shall be provided:

- `bucket` bucket name
- `endpoint` storage endpoint url, e.g. `https://s3.hkg02.cloud-object-storage.appdomain.cloud`
- `apikey` API Key
- `crn` resource instance id

For example, let us create a default configuration:

```
ascli conf preset update mycos --bucket=mybucket
 ↪  --endpoint=https://s3.us-east.cloud-object-storage.appdomain.cloud --apikey=abcdefgh
 ↪  --crn=crn:v1:bluemix:public:iam-identity::a/xxxxxxx
ascli conf preset set default cos mycos
```

Then, jump to the transfer example.

14.2 Using service credential file

If you are the COS administrator and don't have yet the credential: Service credentials are directly created using the IBM cloud Console (web UI). Navigate to:

- → Navigation Menu
- → Resource List
- → Storage
- → Select your storage instance
- → Service Credentials
- → New credentials (Leave default role: Writer, no special options)
- → Copy to clipboard

Then save the copied value to a file, e.g. : `$HOME/cos_service_creds.json`

or using the IBM Cloud CLI:

```
ibmcloud resource service-keys
ibmcloud resource service-key _service_key_name_here_ --output JSON|jq
 ↪  '.[0].credentials'>$HOME/service_creds.json
```

(if you don't have `jq` installed, extract the structure as follows)

It consists in the following structure:

```
{
  "apikey": "my_api_key_here",
  "cos_hmac_keys": {
    "access_key_id": "my_access_key_here",
    "secret_access_key": "my_secret_here"
  },
  "endpoints": "https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints",
  "iam_apikey_description": "my_description_here",
  "iam_apikey_name": "my_key_name_here",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/xxxxxxx.....",
  "resource_instance_id": "crn:v1:bluemix:public:cloud-object-storage:global:a/xxxxxxx....."
}
```

The field `resource_instance_id` is for option `crn`

The field `apikey` is for option `apikey`

(If needed: endpoints for regions can be found by querying the `endpoints` URL.)

The required options for this method are:

- `bucket` bucket name
- `region` bucket region, e.g. eu-de
- `service_credentials` see below

For example, let us create a default configuration:

```
ascli conf preset update mycos --bucket=laurent
 ↪  --service-credentials=@val:@json:@file:~/service_creds.json --region=us-south
ascli conf preset set default cos mycos
```

14.3 Operations, transfers

Let's assume you created a default configuration from once of the two previous steps (else specify the access options on command lines).

A subset of `node` plugin operations are supported, basically node API:

```
ascli cos node info
ascli cos node upload 'faux:///sample1G?1g'
```

> Note: we generate a dummy file `sample1G` of size 2GB using the `faux` PVCL (man `ascp` and section above), but you can of course send a real file by specifying a real file instead.

14.4 COS sample commands

```
cos --bucket=my_bucket_name --endpoint=my_bucket_endpoint --apikey=my_bucket_apikey
 ↪  --crn=my_resource_instance_id node info
cos --bucket=my_bucket_name --region=my_bucket_region
 ↪  --service-credentials=@json:@file:service_creds.json node info
cos node access_key show self
cos node download testfile.bin --to-folder=.
cos node info
cos node upload testfile.bin
```

Chapter 15

IBM Aspera Sync

An interface for the `async` utility is provided in the following plugins:

- server sync
- node sync
- aoc files sync (uses node)
- shares files sync (uses node)

The main advantage over the `async` command line when using `server` is the possibility to use a configuration file, using standard options of `ascli`.

In this case, some of the `sync` parameters are filled by the related plugin using transfer spec parameters (e.g. including token).

> Note: All `sync` commands require an `async` enabled license and availability of the `async` executable (and `asyncadmin`).

Two JSON syntax are supported for option `sync_info`.

15.1 async JSON: API format

It is the same payload as specified on the option `--conf` of `async` or in node API `/asyncs`. This is the preferred syntax and allows a single session definition. But there is no progress output nor error messages.

Documentation on Async node API can be found on IBM Developer Portal.

15.2 async JSON: options mapping

`ascli` defines a JSON equivalent to regular `async`options. It is based on a JSON representation of `async` command line options. It allows definition of multiple sync sessions in a single command, although usually only one sync session is defined.

Chapter 16

Plugin: **preview**: Preview generator for AoC

The `preview` generates thumbnails (office, images, video) and video previews on storage for use primarily in the Aspera on Cloud application. It uses the node API of Aspera HSTS and requires use of Access Keys and its storage root. Several parameters can be used to tune several aspects:

- Methods for detection of new files needing generation
- Methods for generation of video preview
- Parameters for video handling

16.1 Aspera Server configuration

Specify the previews folder as shown in:

https://ibmaspera.com/help/admin/organization/installing_the_preview_maker

By default, the `preview` plugin expects previews to be generated in a folder named `previews` located in the storage root. On the transfer server execute:

```
PATH=/opt/aspera/bin:$PATH

asconfigurator -x "server;preview_dir,previews"
asnodeadmin --reload
```

> Note: the configuration `preview_dir` is relative to the storage root, no need leading or trailing `/`. In general just set the value to `previews`

If another folder is configured on the HSTS, then specify it to `ascli` using the option `previews_folder`.

The HSTS node API limits any preview file to a parameter: `max_request_file_create_size_kb` (1 KB is 1024 bytes). This size is internally capped to 1<<24 Bytes (16777216) , i.e. 16384 KBytes.

To change this parameter in `aspera.conf`, use `asconfigurator`. To display the value, use `asuserdata`:

```
asuserdata -a | grep max_request_file_create_size_kb

  max_request_file_create_size_kb: "1024"

asconfigurator -x "server; max_request_file_create_size_kb,16384"
```

If you use a value different than 16777216, then specify it using option `max_size`.

> Note: the HSTS parameter (`max_request_file_create_size_kb`) is in kiloBytes while the generator parameter is in Bytes (factor of 1024).

16.2 External tools: Linux

`ascli` requires the following external tools available in the `PATH`:

- ImageMagick : `convert composite`
- OptiPNG : `optipng`
- FFmpeg : `ffmpeg ffprobe`
- Libreoffice : `libreoffice`

Here shown on Redhat/CentOS.

Other OSes should work as well, but are note tested.

To check if all tools are found properly, execute:

```
ascli preview check
```

16.2.1 Image: ImageMagick and optipng

```
dnf install -y ImageMagick optipng
```

You may also install `ghostscript` which adds fonts to ImageMagick. Available fonts, used to generate png for text, can be listed with `magick identify -list font`. Prefer ImageMagick version >=7.

More info on ImageMagick at https://imagemagick.org/

16.2.2 Video: FFmpeg

The easiest method is to download and install the latest released version of ffmpeg with static libraries from https://johnvansickle.com/ffmpeg/

```
curl -s https://johnvansickle.com/ffmpeg/releases/ffmpeg-release-amd64-static.tar.xz|(mkdir -p
↪  /opt && cd /opt && rm -f ffmpeg /usr/bin/{ffmpeg,ffprobe} && rm -fr ffmpeg-*-amd64-static &&
↪  tar xJvf - && ln -s ffmpeg-* ffmpeg && ln -s /opt/ffmpeg/{ffmpeg,ffprobe} /usr/bin)
```

16.2.3 Office: Unoconv and Libreoffice

If you don't want to have preview for office documents or if it is too complex you can skip office document preview generation by using option: `--skip-types=office`

The generation of preview in based on the use of `unoconv` and `libreoffice`

- CentOS 8

```
dnf install unoconv
```

- Amazon Linux

```
amazon-linux-extras enable libreoffice
yum clean metadata
yum install libreoffice-core libreoffice-calc libreoffice-opensymbol-fonts libreoffice-ure
↪  libreoffice-writer libreoffice-pyuno libreoffice-impress
wget https://raw.githubusercontent.com/unoconv/unoconv/master/unoconv
mv unoconv /usr/bin
chmod a+x /usr/bin/unoconv
```

16.3 Configuration

The preview generator should be executed as a non-user. When using object storage, any user can be used, but when using local storage it is usually better to use the user `xfer`, as uploaded files are under this identity: this ensures proper access rights. (we will assume this)

Like any `ascli` commands, parameters can be passed on command line or using a configuration option preset. The configuration file must be created with the same user used to run so that it is properly used on runtime.

The `xfer` user has a special protected shell: `aspshell`, so in order to update the configuration, and when changing identity, specify an alternate shell. E.g.:

```
su -s /bin/bash - xfer

ascli config preset update mypreviewconf --url=https://localhost:9092 --username=my_access_key
↪  --password=my_secret --skip-types=office --lock-port=12346

ascli config preset set default preview mypreviewconf
```

Here we assume that Office file generation is disabled, else remove this option. `lock_port` prevents concurrent execution of generation when using a scheduler.

One can check if the access key is well configured using:

```
ascli -Ppreviewconf node browse /
```

This shall list the contents of the storage root of the access key.

16.4 Options for generated files

When generating preview files, some options are provided by default. Some values for the options can be modified on command line. For video preview, the whole set of options can be overridden with option `reencode_ffmpeg`: it is a `Hash` with two keys: `in` and `out`, each is an `Array` of strings with the native options to `ffmpeg`.

16.5 Execution

`ascli` intentionally supports only a one shot mode (no infinite loop) in order to avoid having a hanging process or using too many resources (calling REST api too quickly during the scan or event method). It needs to be run on a regular basis to create or update preview files. For that use your best reliable scheduler, see Scheduler.

Typically, for Access key access, the system/transfer is `xfer`. So, in order to be consistent have generate the appropriate access rights, the generation process should be run as user `xfer`.

Lets do a one shot test, using the configuration previously created:

```
su -s /bin/bash - xfer

ascli preview scan --overwrite=always
```

When the preview generator is first executed it will create a file: `.aspera_access_key` in the previews folder which contains the access key used. On subsequent run it reads this file and check that previews are generated for the same access key, else it fails. This is to prevent clash of different access keys using the same root.

16.6 Configuration for Execution in scheduler

Details are provided in section Scheduler.

Shorter commands can be specified if a configuration preset was created as shown previously.

For example the timeout value can be differentiated depending on the option: event versus scan:

```
case "$*" in *trev*) tmout=10m ;; *) tmout=30m ;; esac
```

16.7 Candidate detection for creation or update (or deletion)

`ascli` generates preview files using those commands:

- `trevents` : only recently uploaded files will be tested (transfer events)
- `events` : only recently uploaded files will be tested (file events: not working)
- `scan` : recursively scan all files under the access key's "storage root"
- `test` : test using a local file

Once candidate are selected, once candidates are selected, a preview is always generated if it does not exist already, else if a preview already exist, it will be generated using one of three values for the `overwrite` option:

- `always` : preview is always generated, even if it already exists and is newer than original
- `never` : preview is generated only if it does not exist already
- `mtime` : preview is generated only if the original file is newer than the existing

Deletion of preview for deleted source files: not implemented yet (TODO).

If the `scan` or `events` detection method is used, then the option : `skip_folders` can be used to skip some folders. It expects a list of path relative to the storage root (docroot) starting with slash, use the `@json:` notation, example:

```
ascli preview scan --skip-folders=@json:'["/not_here"]'
```

The option `folder_reset_cache` forces the node service to refresh folder contents using various methods.

When scanning the option `query` has the same behavior as for the `node access_keys do self find` command.

Refer to that section for details.

16.8 Preview File types

Two types of preview can be generated:

- `png`: thumbnail
- `mp4`: video preview (only for video)

Use option `skip_format` to skip generation of a format.

16.9 Supported input Files types

The preview generator supports rendering of those file categories:

- image
- pdf
- plaintext
- office
- video

To avoid generation for some categories, specify a list using option `skip_types`.

Each category has a specific rendering method to produce the png thumbnail.

The mp4 video preview file is only for category `video`

File type is primarily based on file extension detected by the node API and translated info a mime type returned by the node API.

16.10 mimemagic

By default, the Mime type used for conversion is the one returned by the node API, based on file name extension.

It is also possible to detect the mime type using option `mimemagic`. To use it, set option `mimemagic` to `yes`: `--mimemagic=yes`.

This requires to manually install the mimemagic gem: `gem install mimemagic`.

In this case the `preview` command will first analyze the file content using mimemagic, and if no match, will try by extension.

If the `mimemagic` gem complains about missing mime info file:

- any OS:
  - Examine the error message
  - Download the file: freedesktop.org.xml.in
  - move and rename this file to one of the locations expected by mimemagic as specified in the error message
- Windows:

  - Download the file: freedesktop.org.xml.in
  - Place this file in the root of Ruby (or elsewhere): `C:\RubyVV-x64\freedesktop.org.xml.in`
  - Set a global variable using `SystemPropertiesAdvanced.exe` or using `cmd` (replace VV with version) to the exact path of this file:

```
SETX FREEDESKTOP_MIME_TYPES_PATH C:\RubyVV-x64\freedesktop.org.xml.in
```

  - Close the `cmd` and restart a new one if needed to get refreshed env vars

- Linux RHEL 8+:

```
dnf install shared-mime-info
```

- macOS:

```
brew install shared-mime-info
```

16.11 Generation: Read source files and write preview

Standard open source tools are used to create thumbnails and video previews. Those tools require that original files are accessible in the local file system and also write generated files on the local file system. `ascli` provides 2 ways to read and write files with the option: `file_access`

If the preview generator is run on a system that has direct access to the file system, then the value `local` can be used. In this case, no transfer happen, source files are directly read from the storage, and preview files are directly written to the storage.

If the preview generator does not have access to files on the file system (it is remote, no mount, or is an object storage), then the original file is first downloaded, then the result is uploaded, use method `remote`.

16.12 Preview sample commands

```
preview check --skip-types=office
preview scan --scan-id=1 --skip-types=office --log-level=info --file-access=remote
 ↪  --ts=@json:'{"target_rate_kbps":1000000}'
preview scan --skip-types=office --log-level=info
preview show --base=test my_docx
preview show --base=test my_mpg --video-png-conv=animated
preview show --base=test my_mpg --video-png-conv=fixed
preview show --base=test my_mpg mp4 --video-conversion=clips
preview show --base=test my_mpg mp4 --video-conversion=reencode
preview show --base=test my_pdf
preview test --base=test my_dcm
preview test --base=test my_mxf mp4 --video-conversion=blend
 ↪  --query=@json:'{"text":true,"double":true}'
preview trevents --once-only=yes --skip-types=office --log-level=info
```

Chapter 17

SMTP for email notifications

`ascli` can send email, for that setup SMTP configuration. This is done with option `smtp`.

The `smtp` option is a `Hash` (extended value) with the following fields:

field	default	example	description
`server`	-	smtp.gmail.com	SMTP server address
`tls`	true	true	enable STARTTLS (port 587)
`ssl`	false	false	enable TLS (port 465)
`port`	587 or 465 or 25	587	port for service
`domain`	domain of server	gmail.com	email domain of user
`username`	-	john@example.com	user to authenticate on SMTP server, leave empty for open auth.
`password`	-	my_password_here	password for above username
`from_email`	username if defined	johnny@example.com	address used if receiver replies
`from_name`	same as email	John Wayne	display name of sender

17.1 Example of configuration

```
ascli config preset set smtp_google server smtp.google.com
ascli config preset set smtp_google username john@gmail.com
ascli config preset set smtp_google password my_password_here
```

or

```
ascli config preset init smtp_google @json:'{"server":"smtp.google.com","username":"john@gmail.
↪  com","password":"my_password_here"}'
```

or

```
ascli config preset update smtp_google --server=smtp.google.com --username=john@gmail.com
↪  --password=my_password_here
```

Set this configuration as global default, for instance:

```
ascli config preset set cli_default smtp @val:@preset:smtp_google
ascli config preset set default config cli_default
```

17.2 Email templates

Sent emails are built using a template that uses the ERB syntax.

The template is the full SMTP message, including headers.

The following variables are defined by default:

- from_name
- from_email
- to

Other variables are defined depending on context.

17.3 Test

Check settings with `smtp_settings` command. Send test email with `email_test`.

```
ascli config --smtp=@preset:smtp_google smtp
ascli config --smtp=@preset:smtp_google email --notif-to=sample.dest@example.com
```

17.4 Notifications for transfer status

An e-mail notification can be sent upon transfer success and failure (one email per transfer job, one job being possibly multi session, and possibly after retry).

To activate, use option `notif_to`.

A default e-mail template is used, but it can be overridden with option `notif_template`.

The environment provided contains the following additional variables:

- subject
- body
- global_transfer_status
- ts

Example of template:

```
From: <%=from_name%> <<%=from_email%>>
To: <<%=to%>>
Subject: <%=subject%>

Transfer is: <%=global_transfer_status%>
```

Chapter 18

Tool: **asession**

This gem comes with a second executable tool providing a simplified standardized interface to start a FASP session: `asession`.

It aims at simplifying the startup of a FASP session from a programmatic stand point as formatting a transfer-spec is:

- common to Aspera Node API (HTTP POST /ops/transfer)
- common to Aspera Connect API (browser javascript startTransfer)
- easy to generate by using any third party language specific JSON library

Hopefully, IBM integrates this directly in `ascp`, and this tool is made redundant.

This makes it easy to integrate with any language provided that one can spawn a sub process, write to its STDIN, read from STDOUT, generate and parse JSON.

`ascli` expect one single argument: a transfer-spec.

If no argument is provided, it assumes a value of: `@json:@stdin:`, i.e. a JSON formatted transfer-spec on stdin.

> Note: If JSON is the format, specify `@json:` to tell `ascli` to decode the `Hash` using JSON syntax.

During execution, it generates all low level events, one per line, in JSON format on stdout.

There are special ”extended” transfer-spec parameters supported by `asession`:

- `EX_loglevel` to change log level of `ascli`
- `EX_file_list_folder` to set the folder used to store (exclusively, because of garbage collection) generated file lists. By default it is `[system tmp folder]/[username]_asession_filelists`

> Note: In addition, many ”EX_” transfer-spec parameters are supported for the `direct` transfer agent (used by `asession`), refer to section transfer-spec.

18.1 Comparison of interfaces

feature/tool	asession	`ascp`	FaspManager	Transfer SDK
language integration	any	any	C/C++C#/.netGoPy-thonjava	many
required additional components to `ascp`	RubyAspera	-	library(headers)	daemon
startup	JSON on stdin(standard APIs:JSON.gener-atePro-cess.spawn)	command line arguments	API	daemon

feature/tool	asession	ascp	FaspManager	Transfer SDK
events	JSON on stdout	none by defaultor need to open management portand proprietary text syntax	callback	callback
platforms	any with Ruby and ascp	any with ascp (and SDK if compiled)	any with ascp	any with ascp and transfer daemon

18.2 Simple session

Create a file `session.json` with:

```
{"remote_host":"demo.asperasoft.com","remote_user":"asperaweb","ssh_port":33001,
↪  "remote_password":"my_password_here","direction":"receive","destination_root":"./test.dir",
↪  "paths":[{"source":"/aspera-test-dir-tiny/200KB.1"}],"resume_level":"none"}
```

Then start the session:

```
asession < session.json
```

18.3 Asynchronous commands and Persistent session

`asession` also supports asynchronous commands (on the management port). Instead of the traditional text protocol as described in `ascp` manual, the format for commands is: one single line per command, formatted in JSON, where parameters shall be "snake" style, for example: `LongParameter` -> `long_parameter`

This is particularly useful for a persistent session (with the transfer-spec parameter: `"keepalive":true`)

```
asession
{"remote_host":"demo.asperasoft.com","ssh_port":33001,"remote_user":"asperaweb",
↪  "remote_password":"my_password_here","direction":"receive","destination_root":".",
↪  "keepalive":true,"resume_level":"none"}
{"type":"START","source":"/aspera-test-dir-tiny/200KB.2"}
{"type":"DONE"}
```

(events from FASP are not shown in above example. They would appear after each command)

18.4 Example of language wrapper

Nodejs: https://www.npmjs.com/package/aspera

18.5 Help

```
asession -h
USAGE
    asession
    asession -h|--help
    asession <transfer spec extended value>

    If no argument is provided, default will be used: @json:@stdin
    -h, --help display this message
    <transfer spec extended value> a JSON value for transfer_spec, using the prefix: @json:
    The value can be either:
       the JSON description itself, e.g. @json:'{"xx":"yy",...}'
```

```
      @json:@stdin, if the JSON is provided from stdin
      @json:@file:<path>, if the JSON is provided from a file
   Asynchronous commands can be provided on STDIN, examples:
      {"type":"START","source":"/aspera-test-dir-tiny/200KB.2"}
      {"type":"START","source":"xx","destination":"yy"}
      {"type":"DONE"}
Note: debug information can be placed on STDERR, using the "EX_loglevel" parameter in transfer
 ↪  spec (debug=0)
EXAMPLES
   asession @json:'{"remote_host":"demo.asperasoft.com","remote_user":"asperaweb","ssh_port":
    ↪  33001,"remote_password":"demoaspera","direction":"receive","destination_root":"./test.
    ↪  dir","paths":[{"source":"/aspera-test-dir-tiny/200KB.1"}]}'
   echo '{"remote_host":...}'|asession @json:@stdin
```

Chapter 19

Hot folder

19.1 Requirements

`ascli` maybe used as a simple hot folder engine. A hot folder being defined as a tool that:

- locally (or remotely) detects new files in a top folder
- send detected files to a remote (respectively, local) repository
- only sends new files, do not re-send already sent files
- optionally: sends only files that are not still ”growing”
- optionally: after transfer of files, deletes or moves to an archive

In addition: the detection should be made ”continuously” or on specific time/date.

19.2 Setup procedure

The general idea is to rely on :

- existing `ascp` features for detection and transfer
- take advantage of `ascli` configuration capabilities and server side knowledge
- the OS scheduler for reliability and continuous operation

19.2.1 **ascp** features

Interesting `ascp` features are found in its arguments: (see `ascp` manual):

- `ascp` already takes care of sending only new files: option `-k 1,2,3` (`resume_policy`)
- `ascp` has some options to remove or move files after transfer: `--remove-after-transfer`, `--move-after-transfer`, `--remove-empty-directories` (`remove_after_transfer`, `move_after_transfer`, `remove_empty_directories`)
- `ascp` has an option to send only files not modified since the last X seconds: `--exclude-newer-than`, `--exclude-older-than` (`exclude_newer_than`,`exclude_older_than`)
- `--src-base` (`src_base`) if top level folder name shall not be created on destination

  Note: `ascli` takes transfer parameters exclusively as a transfer-spec, with `ts` option.

  Note: Most, but not all, native `ascp` arguments are available as standard transfer-spec parameters.

  Note: Only for the `direct` transfer agent (not others, like connect or node), native `ascp` arguments can be provided with parameter `ascp_args` of option `transfer_info` .

19.2.2 server side and configuration

Virtually any transfer on a ”repository” on a regular basis might emulate a hot folder.

Note: file detection is not based on events (inotify, etc...), but on a simple folder scan on source side.

Note: parameters may be saved in a option preset and used with `-P`.

19.2.3 Scheduling

Once `ascli` parameters are defined, run the command using the OS native scheduler, e.g. every minutes, or 5 minutes, etc... Refer to section Scheduler. (on use of option `lock_port`)

19.3 Example: upload hot folder

```
ascli server upload source_hot --to-folder=/Upload/target_hot --lock-port=12345
↪  --ts=@json:'{"remove_after_transfer":true,"remove_empty_directories":true,
↪  "exclude_newer_than:-8,"src_base":"source_hot"}'
```

The local folder (here, relative path: `source_hot`) is sent (upload) to an aspera server. Source files are deleted after transfer. Growing files will be sent only once they don't grow anymore (based on an 8-second cool-off period). If a transfer takes more than the execution period, then the subsequent execution is skipped (`lock_port`) preventing multiple concurrent runs.

19.4 Example: unidirectional synchronization (upload) to server

```
ascli server upload source_sync --to-folder=/Upload/target_sync --lock-port=12345
↪  --ts=@json:'{"resume_policy":"sparse_csum","exclude_newer_than":-8,"src_base":"source_sync"}'
```

This can also be used with other folder-based applications: Aspera on Cloud, Shares, Node:

19.5 Example: unidirectional synchronization (download) from Aspera on Cloud Files

```
ascli aoc files download . --to-folder=. --lock-port=12345 --progressbar=no --display=data
↪  --ts=@json:'{"resume_policy":"sparse_csum","target_rate_kbps":50000,"exclude_newer_than":-8,
↪  "delete_before_transfer":true}'
```

Note: option `delete_before_transfer` will delete files locally, if they are not present on remote side.

Note: options `progress` and `display` limit output for headless operation (e.g. cron job)

Chapter 20

Health check and Nagios

Most plugin provide a `health` command that will check the health status of the application. Example:

```
ascli console health
```

```
+--------+-------------+------------+
| status | component   | message    |
+--------+-------------+------------+
| ok     | console api | accessible |
+--------+-------------+------------+
```

Typically, the health check uses the REST API of the application with the following exception: the `server` plugin allows checking health by:

- issuing a transfer to the server
- checking web app status with `asctl all:status`
- checking daemons process status

`ascli` can be called by Nagios to check the health status of an Aspera server. The output can be made compatible to Nagios with option `--format=nagios` :

```
ascli server health transfer --to-folder=/Upload --format=nagios --progressbar=no
```

```
OK - [transfer:ok]
```

```
ascli server health asctl status --cmd_prefix='sudo ' --format=nagios
```

```
OK - [NP:running, MySQL:running, Mongrels:running, Background:running, DS:running, DB:running,
↪  Email:running, Apache:running]
```

Chapter 21

Ruby Module: **`Aspera`**

Main components:

- `Aspera` generic classes for REST and OAuth
- `Aspera::Fasp`: starting and monitoring transfers. It can be considered as a FASPManager class for Ruby.
- `Aspera::Cli`: `ascli`.

Working examples can be found in repo: https://github.com/laurent-martin/aspera-api-examples in Ruby examples.

Chapter 22

Changes (Release notes)

See CHANGELOG.md

Chapter 23

History

When I joined Aspera, there was only one CLI: `ascp`, which is the implementation of the FASP protocol, but there was no CLI to access the various existing products (Server, Faspex, Shares). Once, Serban (founder) provided a shell script able to create a Faspex Package using Faspex REST API. Since all products relate to file transfers using FASP (`ascp`), I thought it would be interesting to have a unified CLI for transfers using FASP. Also, because there was already the `ascp` tool, I thought of an extended tool : `eascp.pl` which was accepting all `ascp` options for transfer but was also able to transfer to Faspex and Shares (destination was a kind of URI for the applications).

There were a few pitfalls:

- `ascli` was written in the aging `perl` language while most Aspera web application products (but the Transfer Server) are written in `ruby`.
- `ascli` was only for transfers, but not able to call other products APIs

So, it evolved into `ascli`:

- portable: works on platforms supporting `ruby` (and `ascp`)
- easy to install with the `gem` utility
- supports transfers with multiple Transfer Agents, that's why transfer parameters moved from `ascp` command line to transfer-spec (more reliable , more standard)
- `ruby` is consistent with other Aspera products

Over the time, a supported command line tool `aspera` was developed in C++, it was later on deprecated. It had the advantage of being relatively easy to installed, as a single executable (well, still using `ascp`), but it was too limited IMHO, and lacked a lot of the features of this CLI.

Enjoy a coffee on me:

```
ascli conf coffee
ascli conf coffee --ui=text
```

Chapter 24

Common problems

24.1 Error: ”Remote host is not who we expected”

Cause: `ascp` >= 4.x checks fingerprint of highest server host key, including ECDSA. `ascp` < 4.0 (3.9.6 and earlier) support only to RSA level (and ignore ECDSA presented by server). `aspera.conf` supports a single fingerprint.

Workaround on client side: To ignore the certificate (SSH fingerprint) add option on client side (this option can also be added permanently to the config file):

```
--ts=@json:'{"sshfp":null}'
```

Workaround on server side: Either remove the fingerprint from `aspera.conf`, or keep only RSA host keys in `sshd_config`.

References: ES-1944 in release notes of 4.1 and to HSTS admin manual section ”Configuring Transfer Server Authentication With a Host-Key Fingerprint”.

24.2 Error ”can't find header files for ruby”

Some Ruby gems dependencies require compilation of native parts (C). This also requires Ruby header files. If Ruby was installed as a Linux Packages, then also install Ruby development package: `ruby-dev` ir `ruby-devel`, depending on distribution.

24.3 ED255519 key not supported

ED25519 keys are deactivated since version 0.9.24 so this type of key will just be ignored.

Without this deactivation, if such key was present the following error was generated:

```
OpenSSH keys only supported if ED25519 is available
```

Which meant that you do not have Ruby support for ED25519 SSH keys. You may either install the suggested Gems, or remove your ed25519 key from your `.ssh` folder to solve the issue.