## Command line clients

The *apache-sshd.zip* distribution provides Windows/Linux scripts that use the MINA SSHD code base to implement the common *ssh, scp, sftp* commands. The clients accept most useful switches from the original commands they mimic, where the -o Option=Value arguments can be used to configure the client/server in addition to the system properties mechanism. For more details, consult the *main* methods code in the respective SshClientMain, SftpCommandMain and ScpClientMain classes. The code also includes SshKeyScanMain that is a simple implementation for [ssh-keyscan(1)](https://www.freebsd.org/cgi/man.cgi?query=ssh-keyscan&sektion=1).

The distribution also includes also an *sshd* script that can be used to launch a server instance - see SshServerMain#main for activation command line arguments and options.

In order to use this CLI code as part of another project, one needs to include the *sshd-cli* module:

<dependency>  
 <groupId>org.apache.sshd</groupId>  
 <artifactId>sshd-cli</artifactId>  
 <version>...same version as the core...</version>  
 </dependency>

In general, the CLI clients accept most of their Linux counterpart arguments. Furthermore, one can use the -o Option=Value argument in order to provide **internal** SSHD code configurations (in addition to the ones specified as system properties via -Dprop=value JVM option.

### SftpCommandMain

A CLI client reminiscent of [sftp(1)](https://linux.die.net/man/1/sftp). By default uses an internal SftpClientFactory. This can be overridden as follows:

1. Provide a -o SftpClientFactory=XXX command line argument where the option specifies the fully-qualified name of the class that implements this interface.
2. Add a META-INF\services\org.apache.sshd.sftp.client.SftpClientFactory file containing the fully-qualified name of the class that implements this interface. **Note:** if more than one such instance is detected an exception is thrown.

**Note:** The specified class(es) must be public and contain a public no-args constructor.

The CLI client provides a few extra "commands" that can be used to view metadata information about the current session

* session - Show current SSH session details - including ID, client/server identification line, peer, etc..
* kex - Show KEX details - client proposal, server one and negotiated parameters.
* info - General details about the SFTP protocol - e.g., supported extensions by the server.
* version - The negotiated SFTP protocol version.
* help - List all available commands.
* exit - Quit the SFTP session

### SshClientMain

A CLI client compatible with the [ssh(1)](https://linux.die.net/man/1/ssh) command line options, with a few extra options:

* -io - select a specific IoServiceFactoryFactory:

java -cp ... org.apache.sshd.cli.client.SshClientMain -io <value>

Where value can be:

\* One of the default builtin values (NIO2, MINA, NETTY)  
  
\* A fully qualified class name implementing this interface  
  
If no specific value provided NIO2 is used.

* -w <password> - provide a password as part of the command instead of waiting to be prompted.

java -cp ... org.apache.sshd.cli.client.SshClientMain -l <login> -w <password> ...host...

* SetEnv/SendEnv - can be used to send specific environment variables to the server when executing a command or opening a shell. Example:

java -cp ... org.apache.sshd.cli.client.SshClientMain -o SetEnv=X=7,Y=8  
  
 # Can also be used as separate options  
  
 java -cp ... org.apache.sshd.cli.client.SshClientMain -o SetEnv=X=7 -o SetEnv=Y=8

* RequestTTY - can be no, yes or auto (default). If auto the CLI client will attempt to initialize the PTY options according to the O/S. In **addition** to the auto-detected PTY modes, one can override them by using the PtyMode option:

java -cp ... org.apache.sshd.cli.client.SshClientMain -o PtyMode=VINTR,TTY\_OP\_ISPEED=4200  
  
 # Can also be used as separate options  
  
 java -cp ... org.apache.sshd.cli.client.SshClientMain -o PtyMode=VINTR -o PtyMode=TTY\_OP\_ISPEED=4200

Any option that does not have a specific value specified for it is assumed to use 1 - therefore, in order to **disable** an option one must use -o PtyMode=WHATEVER=0.

### ScpCommandMain

Reminiscent of the [scp(1)](https://man7.org/linux/man-pages/man1/scp.1.html) CLI client - including support for "3-way" copy (a.k.a. remote-to-remote) option:

scp -p -r -3 user1@server1:source user2@server2:destination

In this context, it is worth mentioning that the CLI also supports URI locations having the format scp://[user@]host[:port][/path]

# If port is omitted then 22 is assumed  
scp -p scp://user1@server1:2222/source/file /home/user2/destination  
  
# Note: same effect can be achieved with -P option  
  
scp -p -P 2222 user1@server1:source/file /home/user2/destination  
  
# the URI is better suited for remote-to-remote transfers  
scp -p -r -3 scp://user1@server1:2222/source scp://user2@server2:3333/destination

### SshServerMain

Command line SSH daemon

* **Port** - by default the SSH server sets up to list on port 8000 in order to avoid conflicts with any running SSH O/S daemon. This can be modified by providing a -p NNNN or -o Port=NNNN command line option.
* **Subsystem(s)** - the server automatically detects subsystems using the [Java ServiceLoader mechanism](https://docs.oracle.com/javase/8/docs/api/java/util/ServiceLoader.html).

This can be overwritten as follows (in this order):

1. Provide a org.apache.sshd.server.subsystem.SubsystemFactory system property containing comma-separated fully-qualified names of classes implementing this interface. The implementations must be public and have a public no-args constructor for instantiating them. The order of the provided subsystems will be according to their order in the specified list.
2. Provide a -o Subsystem=xxx,yyy command line argument where value is a comma-separated list of the **name**(s) of the auto-detected factories via the ServiceLoader mechanism. The special value none may be used to indicate that no subsystem is to be configured. **Note:** no specific order is provided when subsystems are auto-detected and/or filtered.

* **Shell** - unless otherwise instructed, the default SSH server uses an internal shell (see InteractiveProcessShellFactory). The shell can be overridden or disabled by specifying a -o ShellFactory=XXX option where the value can either be none to specify that no shell is to be used, or the fully-qualified name of a class that implements the ShellFactory interface. The implementation must be public and have a public no-args constructor for instantiating it.

**Note:** A special value of scp can be used to use the built-in ScpShell instead of the interactive one (reminder: the SCP "shell" is a limited shell that provides a good enough functionality for *WinScp*).