

---

# ssh2-python Documentation

*Release 0+unknown*

**P Kittenis**

**Nov 15, 2020**



# CONTENTS:

|          |                                    |           |
|----------|------------------------------------|-----------|
| <b>1</b> | <b>Design And Goals</b>            | <b>3</b>  |
| <b>2</b> | <b>Installation</b>                | <b>5</b>  |
| 2.1      | Pip Binary Packages . . . . .      | 5         |
| 2.2      | System Binary Packages . . . . .   | 5         |
| 2.3      | Conda package . . . . .            | 6         |
| 2.4      | Installation from Source . . . . . | 6         |
| 2.5      | Testing Installation . . . . .     | 7         |
| <b>3</b> | <b>API Documentation</b>           | <b>9</b>  |
| 3.1      | ssh2.session . . . . .             | 9         |
| 3.2      | ssh2.channel . . . . .             | 15        |
| 3.3      | ssh2.agent . . . . .               | 19        |
| 3.4      | ssh2.sftp . . . . .                | 19        |
| 3.5      | ssh2.sftp_handle . . . . .         | 23        |
| 3.6      | ssh2.pkey . . . . .                | 26        |
| 3.7      | ssh2.listener . . . . .            | 26        |
| 3.8      | ssh2.knownhost . . . . .           | 26        |
| 3.9      | Exceptions . . . . .               | 29        |
| 3.10     | Stat Info . . . . .                | 34        |
| 3.11     | File Info . . . . .                | 35        |
| 3.12     | Utility Functions . . . . .        | 35        |
| <b>4</b> | <b>Change Log</b>                  | <b>37</b> |
| 4.1      | 0.18.0 . . . . .                   | 37        |
| 4.2      | 0.17.0.post2 . . . . .             | 37        |
| 4.3      | 0.17.0.post1 . . . . .             | 38        |
| 4.4      | 0.17.0 . . . . .                   | 38        |
| 4.5      | 0.16.0 . . . . .                   | 38        |
| 4.6      | 0.15.0 . . . . .                   | 38        |
| 4.7      | 0.14.0 . . . . .                   | 39        |
| 4.8      | 0.13.0 . . . . .                   | 39        |
| 4.9      | 0.11.0 . . . . .                   | 40        |
| 4.10     | 0.10.0 . . . . .                   | 40        |
| 4.11     | 0.9.1 . . . . .                    | 40        |
| 4.12     | 0.9.0 . . . . .                    | 41        |
| 4.13     | 0.8.0 . . . . .                    | 41        |
| 4.14     | 0.7.0 . . . . .                    | 41        |
| 4.15     | 0.6.0 . . . . .                    | 41        |
| 4.16     | 0.5.5 . . . . .                    | 42        |

|          |                            |           |
|----------|----------------------------|-----------|
| 4.17     | 0.5.4                      | 42        |
| 4.18     | 0.5.3                      | 42        |
| 4.19     | 0.5.2                      | 43        |
| 4.20     | 0.5.1                      | 43        |
| 4.21     | 0.5.0                      | 43        |
| 4.22     | 0.4.0                      | 43        |
| 4.23     | 0.3.1                      | 44        |
| 4.24     | 0.3.0                      | 44        |
| <b>5</b> | <b>Indices and tables</b>  | <b>45</b> |
|          | <b>Python Module Index</b> | <b>47</b> |
|          | <b>Index</b>               | <b>49</b> |

Super fast SSH2 protocol library. `ssh2-python` provides Python bindings for `libssh2`.



## DESIGN AND GOALS

This project's goals are to map 100% of the `libssh2` C API to Python, using Python semantics where appropriate.

Design wise, the library is intentionally a thin wrapper of `libssh2`, implemented in Cython, in order to have as little overhead and conversely as high performance as possible.

Contributions are most welcome!





## INSTALLATION

The recommended installation method is `pip`.

### 2.1 Pip Binary Packages

Binary wheel packages are provided for Linux, OSX and Windows, all Python versions, with `libssh2` and its dependencies included.

Wheel packages have **no dependencies**.

`pip` may need to be updated to be able to install binary wheel packages.

```
pip install -U pip
pip install ssh2-python
```

---

**Note:** Latest available version of OpenSSL at the time the package is built is included in binary wheel packages.

To control which version of OpenSSL is used for the installation either use system packages which use system libraries, the conda package, or install from source.

---

### 2.2 System Binary Packages

System packages can be built for Centos/RedHat 7, Ubuntu 14.04/16.04/18.04, Debian 8 and Fedora 22/23/24 by running `ci/docker/build-packages.sh` script in the repository's directory, based on Docker.

To use the built packages, install via the system's package manager, for example for Centos/RedHat based systems:

```
yum install -y python-ssh2-python-<version>-1.e17.x86_64.rpm
```

---

**Note:** System packages as built by the above script use system provided `libssh2` and do not have all features enabled as most distributions do not have a new enough version. In addition, there are known issues with older versions of `libssh2` like what is provided by distributions.

For best compatibility, it is recommended to install binary packages with `pip`.

---

## 2.3 Conda package

A conda package is available in the conda-forge channel.

To install, run the following.

```
conda install -c conda-forge ssh2-python
```

## 2.4 Installation from Source

Source distributions include a bundled libssh2 which is built automatically by default. OpenSSL development libraries are required.

For builds against system provided libssh2, the `SYSTEM_LIBSSH2=1` environment variable setting can be used.

### 2.4.1 Standard build

Source distributions include a bundled libssh2 which is used by default.

```
git clone git@github.com:ParallelSSH/ssh2-python.git
virtualenv my_env
source my_env/bin/activate
python setup.py install
```

### 2.4.2 System library build

Building against system provided libssh2 is another option which may be preferred. This can be done by setting the `SYSTEM_LIBSSH2=1` environment variable:

```
git clone git@github.com:ParallelSSH/ssh2-python.git
virtualenv my_env
source my_env/bin/activate
export SYSTEM_LIBSSH2=1
python setup.py install
```

### 2.4.3 Custom Compiler Configuration

If there are multiple libssh2 installations on the system, the following can be used to set the include path, runtime and build time library directory paths respectively:

```
git clone git@github.com:ParallelSSH/ssh2-python.git
virtualenv my_env
source my_env/bin/activate
python setup.py build_ext -I /usr/local/include -R /usr/local/lib/x86_64-linux-gnu -L
↪ /usr/local/lib/x86_64-linux-gnu
python setup.py install
```

## Ubuntu

Example for Debian or Ubuntu based distributions.

```
sudo apt-get install libssh2-1-dev python-dev
virtualenv my_env
source my_env/bin/activate
export SYSTEM_LIBSSH2=1
python setup.py install
```

## RedHat

Example for RedHat based distributions.

```
sudo yum install libssh2-devel python-devel
virtualenv my_env
source my_env/bin/activate
export SYSTEM_LIBSSH2=1
python setup.py install
```

## 2.5 Testing Installation

Importing the library should exit without error if installation is successful.

```
python -c 'from ssh2.session import Session'
echo $?
```

**Output** 0



## API DOCUMENTATION

### 3.1 ssh2.session

**class** `ssh2.session.Session`

LibSSH2 Session class providing session functions

**agent\_auth** (*self*, *username*)

Convenience function for performing user authentication via SSH Agent.

Initialises, connects to, gets list of identities from and attempts authentication with each identity from SSH agent.

Note that agent connections cannot be used in non-blocking mode - clients should call *set\_blocking(0)* after calling this function.

On completion, or any errors, agent is disconnected and resources freed.

All steps are performed in C space which makes this function perform better than calling the individual Agent class functions from Python.

**Raises** `MemoryError` on error initialising agent

**Raises** `ssh2.exceptions.AgentConnectionError` on error connecting to agent

**Raises** `ssh2.exceptions.AgentListIdentitiesError` on error getting identities from agent

**Raises** `ssh2.exceptions.AgentAuthenticationError` on no successful authentication with all available identities.

**Raises** `ssh2.exceptions.AgentGetIdentityError` on error getting known identity from agent

**Return type** `None`

**agent\_init** (*self*)

Initialise SSH agent.

**Return type** `ssh2.agent.Agent`

**block\_directions** (*self*)

Get blocked directions for the current session.

From libssh2 documentation:

Can be a combination of:

`ssh2.session.LIBSSH2_SESSION_BLOCK_INBOUND`: Inbound direction blocked.

`ssh2.session.LIBSSH2_SESSION_BLOCK_OUTBOUND`: Outbound direction blocked.

Application should wait for data to be available for socket prior to calling a libssh2 function again. If `LIBSSH2_SESSION_BLOCK_INBOUND` is set `select` should contain the session socket in `readfds` set.

Correspondingly in case of `LIBSSH2_SESSION_BLOCK_OUTBOUND` `writefds` set should contain the socket.

**Return type** `int`

**direct\_tcpip** (*self*, *host*, *int port*)

Open direct TCP/IP channel to `host:port`

Channel will be listening on an available open port on client side as assigned by OS.

**direct\_tcpip\_ex** (*self*, *host*, *int port*, *shost*, *int sport*)

**disconnect** (*self*)

**flag** (*self*, *set\_flag*, *value*)

Set options for the session.

`set_flag` is the option to set, while `value` is typically set to 1 or 0 to enable or disable the option.

Valid flags are:

- **ssh2.session.LIBSSH2\_FLAG\_SIGPIPE** If set, libssh2 will not attempt to block SIGPIPEs but will let them trigger from the underlying socket layer.
- **ssh2.session.LIBSSH2\_FLAG\_COMPRESS** If set - before the connection negotiation is performed - libssh2 will try to negotiate compression enabling for this connection. By default libssh2 will not attempt to use compression.

Must be called before `self.handshake()` if you wish to change options.

**Raises** `ssh2.exceptions.MethodNotSupported` on an incorrect `flag` or `value` argument(s).

**Parameters**

- **set\_flag** (`ssh2.session.LIBSSH2_METHOD_*`) – Flag to set. See above for options.
- **value** – Value that `set_flag` will be set to. Must be 0 or

1. :type value: int :rtype: int

**forward\_listen** (*self*, *int port*)

Create forward listener on port.

**Parameters** `port` (*int*) – Port to listen on.

**Return type** `ssh2.listener.Listener` or `None`

**forward\_listen\_ex** (*self*, *host*, *int port*, *int bound\_port*, *int queue\_maxsize*)

**get\_blocking** (*self*)

Get session blocking mode enabled True/False.

**Return type** `bool`

**get\_timeout** (*self*)

Get current session timeout setting

**handshake** (*self*, *sock*)

Perform SSH handshake.

Must be called after Session initialisation.

**hostkey** (*self*)

Get server host key for this session.

Returns `key`, `key_type` tuple where `key_type` is one of `ssh2.session.LIBSSH2_HOSTKEY_TYPE_RSA`, `ssh2.session.LIBSSH2_HOSTKEY_TYPE_DSS`, or `ssh2.session.LIBSSH2_HOSTKEY_TYPE_UNKNOWN`

**Return type** tuple(bytes, int)

**hostkey\_hash** (*self*, *int hash\_type*)

Get computed digest of the remote system's host key.

**Parameters** `hash_type` (*int*) – One of `ssh2.session.LIBSSH2_HOSTKEY_HASH_MD5` or `ssh2.session.LIBSSH2_HOSTKEY_HASH_SHA1`

**Return type** bytes

**keepalive\_config** (*self*, *bool want\_reply*, *unsigned int interval*)

Configure keep alive settings.

**Parameters**

- **want\_reply** (*bool*) – True/False for reply wanted from server on keep alive messages being sent or not.
- **interval** (*int*) – Required keep alive interval. Set to 0 to disable keepalives.

**keepalive\_send** (*self*)

Send keepalive.

Returns seconds remaining before next keep alive should be sent.

**Return type** int

**knownhost\_init** (*self*)

Initialise a collection of known hosts for this session.

**Return type** `ssh2.knownhost.KnownHost`

**last\_errno** (*self*)

Retrieve last error number from libssh2, if any. Returns 0 on no last error.

**Return type** int

**last\_error** (*self*, *size\_t msg\_size=1024*)

Retrieve last error message from libssh2, if any. Returns empty string on no error message.

**Return type** str

**method\_pref** (*self*, *method\_type*, *pref\_methods*)

Set internal preferences based on `method_type` to `pref_methods`.

Valid `method_type` options are:

- **LIBSSH2\_METHOD\_KEX** For key exchange.
- **LIBSSH2\_METHOD\_HOSTKEY** For selecting host key type.
- **LIBSSH2\_METHOD\_CRYPT\_CS** Encryption between client to server
- **LIBSSH2\_METHOD\_CRYPT\_SC** Encryption between server to client
- **LIBSSH2\_METHOD\_MAC\_CS** MAC between client to server
- **LIBSSH2\_METHOD\_MAC\_SC** MAC between server to client

- **LIBSSH2\_METHOD\_COMP\_CS** Compression between client to server
- **LIBSSH2\_METHOD\_COMP\_SC** Compression between server to client
- **LIBSSH2\_METHOD\_LANG\_CS** Language between client to server
- **LIBSSH2\_METHOD\_LANG\_SC** Language between server to client

Valid options that end in CS are from the client to the server and the inverse is true as well.

Valid `pref_methods` options are dependant on the `method_type` selected. Refer to the libssh2 docs

Must be called before `self.handshake()` if you wish to change the defaults.

Return 0 on success or negative on failure. It returns `ssh2.error_codes.LIBSSH2_ERROR_EAGAIN` when it would otherwise block. While `ssh2.error_codes.LIBSSH2_ERROR_EAGAIN` is a negative number, it isn't really a failure per se.

**Raises** `ssh2.exceptions.MethodNotSupported` on an incorrect `method_type` or `pref_methods` argument(s).

#### Parameters

- **method\_type** (`ssh2.session.LIBSSH2_METHOD_*`) – Method preference to change.
- **pref\_methods** – Comma delimited list as a bytes string of preferred

methods to use with the most preferred listed first and the least preferred listed last. If a method is listed which is not supported by libssh2 it will be ignored and not sent to the remote host during protocol negotiation. :type `pref_methods`: bytes :rtype: int

**methods** (*self*, *method\_type*)

Get internal preferences used to negotiate based on `method_type`.

Valid `method_type` options are:

- **LIBSSH2\_METHOD\_KEX** For key exchange.
- **LIBSSH2\_METHOD\_HOSTKEY** For selecting host key type.
- **LIBSSH2\_METHOD\_CRYPT\_CS** Encryption between client to server
- **LIBSSH2\_METHOD\_CRYPT\_SC** Encryption between server to client
- **LIBSSH2\_METHOD\_MAC\_CS** MAC between client to server
- **LIBSSH2\_METHOD\_MAC\_SC** MAC between server to client
- **LIBSSH2\_METHOD\_COMP\_CS** Compression between client to server
- **LIBSSH2\_METHOD\_COMP\_SC** Compression between server to client
- **LIBSSH2\_METHOD\_LANG\_CS** Language between client to server
- **LIBSSH2\_METHOD\_LANG\_SC** Language between server to client

Valid options that end in CS are from the client to the server and the inverse is true as well.

**Raises** `ssh2.exceptions.MethodNotSupported` on an incorrect `method_type` argument.

**Parameters** **method\_type** (`ssh2.session.LIBSSH2_METHOD_*`) – Method type.

**Return type** bytes

**open\_session** (*self*)

Open new channel session.



**Return type** `ssh2.channel.Channel`

**publickey\_init** (*self*)

Initialise public key subsystem for managing remote server public keys

**scp\_recv** (*self*, *path*)

Receive file via SCP.

Deprecated in favour of `recv2` (requires `libssh2 >= 1.7`).

**Parameters** **path** (*str*) – File path to receive.

**Return type** `tuple(ssh2.channel.Channel, ssh2.statinfo.StatInfo)` or `None`

**scp\_recv2** (*self*, *path*)

Receive file via SCP.

Available only on `libssh2 >= 1.7`.

**Parameters** **path** (*str*) – File path to receive.

**Return type** `tuple(ssh2.channel.Channel, ssh2.fileinfo.FileInfo)` or `None`

**scp\_send** (*self*, *path*, *int mode*, *size\_t size*)

Deprecated in favour of `scp_send64`. Send file via SCP.

**Parameters**

- **path** (*str*) – Local file path to send.
- **mode** (*int*) – File mode.
- **size** (*int*) – size of file

**Return type** `ssh2.channel.Channel`

**scp\_send64** (*self*, *path*, *int mode*, *libssh2\_uint64\_t size*, *time\_t mtime*, *time\_t atime*)

Send file via SCP.

**Parameters**

- **path** (*str*) – Local file path to send.
- **mode** (*int*) – File mode.
- **size** (*int*) – size of file

**Return type** `ssh2.channel.Channel`

**set\_blocking** (*self*, *bool blocking*)

Set session blocking mode on/off.

**Parameters** **blocking** (*bool*) – False for non-blocking, True for blocking. Session default is blocking unless set otherwise.

**set\_last\_error** (*self*, *int errcode*, *errmsg*)

**set\_timeout** (*self*, *long timeout*)

Set the timeout in milliseconds for how long a blocking call may wait until the situation is considered an error and `ssh2.error_codes.LIBSSH2_ERROR_TIMEOUT` is returned.

By default or if timeout set is zero, blocking calls do not time out. :param timeout: Milliseconds to wait before timeout.

**sftp\_init** (*self*)

Initialise SFTP channel.

**Return type** `ssh2.sftp.SFTP`

**startup** (*self*, *sock*)

Deprecated - use self.handshake

**supported\_algs** (*self*, *method\_type*, *algs*)

Get the supported internal preferences based on *method\_type* and *algs*.

Valid *method\_type* options are:

- **LIBSSH2\_METHOD\_KEX** For key exchange.
- **LIBSSH2\_METHOD\_HOSTKEY** For selecting host key type.
- **LIBSSH2\_METHOD\_CRYPT\_CS** Encryption between client to server
- **LIBSSH2\_METHOD\_CRYPT\_SC** Encryption between server to client
- **LIBSSH2\_METHOD\_MAC\_CS** MAC between client to server
- **LIBSSH2\_METHOD\_MAC\_SC** MAC between server to client
- **LIBSSH2\_METHOD\_COMP\_CS** Compression between client to server
- **LIBSSH2\_METHOD\_COMP\_SC** Compression between server to client
- **LIBSSH2\_METHOD\_LANG\_CS** Language between client to server
- **LIBSSH2\_METHOD\_LANG\_SC** Language between server to client

Raises `ssh2.exceptions.MethodNotSupported` on an incorrect *method\_type* or *algs* argument(s).

**Parameters**

- **method\_type** (`ssh2.session.LIBSSH2_METHOD_*`) – Method type.
- **algs** (`bytes str`) – Comma delimited list as a bytes string.

**Return type** `array`

**userauth\_authenticated** (*self*)

True/False for is user authenticated or not.

**Return type** `bool`

**userauth\_hostbased\_fromfile** (*self*, *username*, *privatekey*, *hostname*, *publickey=None*, *passphrase=""*)

**userauth\_keyboardinteractive** (*self*, *username*, *password*)

Perform keyboard-interactive authentication

**Parameters**

- **username** (`str`) – User name to authenticate.
- **password** (`str`) – Password

**userauth\_list** (*self*, *username*)

Retrieve available authentication methods list.

**Return type** `list`

**userauth\_password** (*self*, *username*, *password*)

Perform password authentication

**Parameters**

- **username** (`str`) – User name to authenticate.

- **password** (*str*) – Password

**userauth\_publickey** (*self, username, bytes pubkeydata*)

Perform public key authentication with provided public key data

**Parameters**

- **username** (*str*) – User name to authenticate as
- **pubkeydata** (*bytes*) – Public key data

**Return type** `int`

**userauth\_publickey\_fromfile** (*self, username, privatekey, passphrase="", publickey=None*)

Authenticate with public key from file.

**Return type** `int`

**userauth\_publickey\_frommemory** (*self, username, bytes privatekeyfiledata, passphrase="", bytes publickeyfiledata=None*)

**sock**

## 3.2 ssh2.channel

**class** `ssh2.channel.Channel`

**close** (*self*)

Close channel. Typically done to be able to get exit status.

**eof** (*self*)

Get channel EOF status.

**Return type** `bool`

**execute** (*self, command*)

Execute command.

**Parameters** **command** (*str*) – Command to execute

**Raises** `ssh2.exceptions.ChannelError` on errors executing command

**Return type** `int`

**flush** (*self*)

Flush stdout stream

**flush\_ex** (*self, int stream\_id*)

Flush stream with id

**flush\_stderr** (*self*)

Flush stderr stream

**get\_exit\_signal** (*self*)

Get exit signal, message and language tag, if any, for command.

**Returns** (`returncode`, `exit signal`, `error message`, `language tag`) tuple.

**Return type** `tuple(int, bytes, bytes, bytes)`

**get\_exit\_status** (*self*)

Get exit status of command.

Note that 0 is also failure code for this function.

Best used in non-blocking mode to avoid it being impossible to tell if 0 indicates failure or an actual exit status of 0

**handle\_extended\_data** (*self*, *int ignore\_mode*)

Deprecated, use `handle_extended_data2`

**handle\_extended\_data2** (*self*, *int ignore\_mode*)

**ignore\_extended\_data** (*self*, *int ignore\_mode*)

Deprecated, use `handle_extended_data2`

**poll\_channel\_read** (*self*, *int extended*)

Deprecated - use `session.block_directions` and socket polling instead

**process\_startup** (*self*, *request*, *message=None*)

Startup process on server for request with message.

Request is a supported SSH subsystem and clients would typically use one of `execute/shell/subsystem` functions depending on request type.

#### Parameters

- **request** (*str*) – Request type (`exec/shell/subsystem`).
- **message** (*str* or *None*) – Request message. Content depends on request type and can be *None*.

**pty** (*self*, *term='vt100'*)

Request a PTY (physical terminal emulation) on the channel.

**Parameters** **term** (*str*) – Terminal type to emulate.

**read** (*self*, *size\_t size=1024*)

Read the stdout stream. Returns return code and output buffer tuple.

Return code is the size of the buffer when positive. Negative values are error codes.

**Parameters** **size** (*int*) – Max buffer size to read.

**Return type** (*int*, *bytes*)

**read\_ex** (*self*, *size\_t size=1024*, *int stream\_id=0*)

Read the stream with given id. Returns return code and output buffer tuple.

Return code is the size of the buffer when positive. Negative values are error codes.

**Parameters** **size** (*int*) – Max buffer size to read.

**Return type** (*int*, *bytes*)

**read\_stderr** (*self*, *size\_t size=1024*)

Read the stderr stream. Returns return code and output buffer tuple.

Return code is the size of the buffer when positive. Negative values are error codes.

**Return type** (*int*, *bytes*)

**receive\_window\_adjust** (*self*, *unsigned long adjustment*, *unsigned long force*)

**receive\_window\_adjust2** (*self*, *unsigned long adjustment*, *unsigned long force*)

**send\_eof** (*self*)

Tell the remote host that no further data will be sent on the specified channel. Processes typically interpret this as a closed stdin descriptor.

Returns 0 on success or negative on failure. It returns `LIBSSH2_ERROR_EAGAIN` when it would otherwise block.

**Return type** `int`

**setenv** (*self*, *varname*, *value*)

Set environment variable on channel.

**Parameters**

- **varname** (*str*) – Name of variable to set.
- **value** (*str*) – Value of variable.

**Return type** `int`

**shell** (*self*)

Request interactive shell from channel.

**Raises** `ssh2.exceptions.ChannelError` on errors requesting interactive shell.

**subsystem** (*self*, *subsystem*)

Request subsystem from channel.

**Parameters** **subsystem** (*str*) – Name of subsystem

**wait\_closed** (*self*)

Wait for server to acknowledge channel close command.

**wait\_eof** (*self*)

Wait for the remote end to acknowledge an EOF request.

Returns 0 on success or negative on failure. It returns `ssh2.error_codes.LIBSSH2_ERROR_EAGAIN` when it would otherwise block.

**Return type** `int`

**window\_read** (*self*)**window\_read\_ex** (*self*, *unsigned long read\_avail*, *unsigned long window\_size\_initial*)**window\_write** (*self*)**window\_write\_ex** (*self*, *unsigned long window\_size\_initial*)**write** (*self*, *buf*)

Write buffer to stdin.

Returns tuple of (`return_code`, `bytes_written`).

In blocking mode `bytes_written` will always equal `len(buf)` if no errors have occurred which would raise exception.

In non-blocking mode `return_code` can be `LIBSSH2_ERROR_EAGAIN` and `bytes_written` can be less than `len(buf)`.

Clients should resume from that point on next call to `write`, ie `buf[bytes_written_in_last_call:]`.

---

**Note:** While this function handles unicode strings for `buf` argument, `bytes_written` offset will always be for the *bytes* representation thereof as returned by the C function calls which only handle byte strings.

---

**Parameters** `buf` (*str*) – Buffer to write

**Return type** `tuple(int, int)`

**write\_ex** (*self*, *int stream\_id*, *buf*)

Write buffer to specified stream id.

Returns tuple of (`return_code`, `bytes_written`).

In blocking mode `bytes_written` will always equal `len(buf)` if no errors have occurred which would raise exception.

In non-blocking mode `return_code` can be `LIBSSH2_ERROR_EAGAIN` and `bytes_written` can be less than `len(buf)`.

Clients should resume from that point on next call to the function, ie `buf[bytes_written_in_last_call:]`.

---

**Note:** While this function handles unicode strings for `buf` argument, `bytes_written` offset will always be for the *bytes* representation thereof as returned by the C function calls which only handle byte strings.

---

**Parameters**

- `stream_id` (*int*) – Id of stream to write to
- `buf` (*str*) – Buffer to write

**Return type** `tuple(int, int)`

**write\_stderr** (*self*, *buf*)

Write buffer to stderr.

Returns tuple of (`return_code`, `bytes_written`).

In blocking mode `bytes_written` will always equal `len(buf)` if no errors have occurred which would raise exception.

In non-blocking mode `return_code` can be `LIBSSH2_ERROR_EAGAIN` and `bytes_written` can be less than `len(buf)`.

Clients should resume from that point on next call to write, ie `buf[bytes_written_in_last_call:]`.

---

**Note:** While this function handles unicode strings for `buf` argument, `bytes_written` offset will always be for the *bytes* representation thereof as returned by the C function calls which only handle byte strings.

---

**Parameters** `buf` (*str*) – Buffer to write

**Return type** `tuple(int, int)`

**x11\_req** (*self*, *int* *screen\_number*)

**x11\_req\_ex** (*self*, *int* *single\_connection*, *const char* \**auth\_proto*, *const char* \**auth\_cookie*, *int* *screen\_number*)

**session**

Originating session.

### 3.3 ssh2.agent

**class** `ssh2.agent.Agent`

**connect** (*self*)

Connect to agent.

**Raises** `ssh2.exceptions.AgentConnectionError` on errors connecting to agent.

**Return type** `int`

**disconnect** (*self*)

Disconnect from agent.

**Return type** `int`

**get\_identities** (*self*)

List and get identities from agent

**Return type** `list(ssh2.pkey.PublicKey)`

**list\_identities** (*self*)

This method is a no-op - use `ssh2.agent.Agent.get_identities()` to list and retrieve identities.

**userauth** (*self*, *username*, *PublicKey* *pkey*)

Perform user authentication with specific public key

**Parameters**

- **username** (*str*) – User name to authenticate as
- **pkey** (`ssh2.pkey.PublicKey`) – Public key to authenticate with

**Raises** `ssh2.exceptions.AgentAuthenticationError` on errors authenticating.

**Return type** `int`

### 3.4 ssh2.sftp

SFTP channel class and related SFTP flags.

### 3.4.1 File types

**var LIBSSH2\_SFTP\_S\_IFMT** Type of file mask  
**var LIBSSH2\_SFTP\_S\_IFIFO** Named pipe (fifo)  
**var LIBSSH2\_SFTP\_S\_IFCHR** Character special (character device)  
**var LIBSSH2\_SFTP\_S\_IFDIR** Directory  
**var LIBSSH2\_SFTP\_S\_IFBLK** Block special (block device)  
**var LIBSSH2\_SFTP\_S\_IFREG** Regular file  
**var LIBSSH2\_SFTP\_S\_IFLNK** Symbolic link  
**var LIBSSH2\_SFTP\_S\_IFSOCK** Socket

### 3.4.2 File transfer flags

**var LIBSSH2\_FXF\_READ** File read flag  
**var LIBSSH2\_FXF\_WRITE** File write flag  
**var LIBSSH2\_FXF\_APPEND** File append flag  
**var LIBSSH2\_FXF\_CREAT** File create flag  
**var LIBSSH2\_FXF\_TRUNC** File truncate flag  
**var LIBSSH2\_FXF\_EXCL** Exclusive file flag

### 3.4.3 File mode masks

#### Owner masks

**var LIBSSH2\_SFTP\_S\_IRWXU** Read/write/execute  
**var LIBSSH2\_SFTP\_S\_IRUSR** Read  
**var LIBSSH2\_SFTP\_S\_IWUSR** Write  
**var LIBSSH2\_SFTP\_S\_IXUSR** Execute

#### Group masks

**var LIBSSH2\_SFTP\_S\_IRWXG** Read/write/execute  
**var LIBSSH2\_SFTP\_S\_IRGRP** Read  
**var LIBSSH2\_SFTP\_S\_IWUSR** Write  
**var LIBSSH2\_SFTP\_S\_IXUSR** Execute



## Other masks

**var LIBSSH2\_SFTP\_S\_IRWXO** Read/write/execute

**var LIBSSH2\_SFTP\_S\_IROTH** Read

**var LIBSSH2\_SFTP\_S\_IWOTH** Write

**var LIBSSH2\_SFTP\_S\_IXOTH** Execute

## Generic mode masks

**var LIBSSH2\_SFTP\_ST\_RDONLY** Read only

**var LIBSSH2\_SFTP\_ST\_NOSUID** No suid

**class** `ssh2.sftp.SFTP`  
SFTP session.

**Parameters** `session` (`ssh2.session.Session` pointer) – Session that initiated SFTP.

**get\_channel** (`self`)  
Get new channel from the SFTP session

**last\_error** (`self`)  
Get last error code from SFTP channel.

**Return type** `int`

**lstat** (`self, path`)  
Link stat a file.

**mkdir** (`self, path, long mode`)  
Make directory.

**Parameters**

- **path** (`str`) – Path of directory to create.
- **mode** (`int`) – Permissions mode of new directory.

**Return type** `int`

**Raises** Appropriate exception from `ssh2.exceptions` on errors.

**open** (`self, filename, unsigned long flags, long mode`)  
Open file handle for file name.

**Parameters**

- **filename** (`str`) – Name of file to open.
- **flags** (`int`) – One or more `LIBSSH2_FXF_*` flags.  
Eg for reading flags is `LIBSSH2_FXF_READ`,  
for writing `LIBSSH2_FXF_WRITE`,  
for both `LIBSSH2_FXF_READ | LIBSSH2_FXF_WRITE`.
- **mode** (`int`) – File permissions mode. `LIBSSH2_SFTP_S_IRUSR` for reading.  
For writing one or more `LIBSSH2_SFTP_S_*` flags.  
Eg, for 664 permission mask (read/write owner/group, read other),  
mode is

```
LIBSSH2_SFTP_S_IRUSR | LIBSSH2_SFTP_S_IWUSR | \  
LIBSSH2_SFTP_S_IRGRP | LIBSSH2_SFTP_S_IWGRP | \  
LIBSSH2_SFTP_S_IROTH
```

**Raises** `ssh2.exceptions.SFTPHandleError` on errors opening file.

**open\_ex** (*self*, *const char \*filename*, *unsigned int filename\_len*, *unsigned long flags*, *long mode*, *int open\_type*)

**opendir** (*self*, *path*)

Open handle to directory path.

**Parameters** **path** (*str*) – Path of directory

**Return type** `ssh2.sftp.SFTPHandle` or `None`

**Raises** `ssh2.exceptions.SFTPHandleError` on errors opening directory.

**realpath** (*self*, *path*, *size\_t max\_len=256*)

Get real path for path.

**Param** Path name to get real path for.

**Parameters** **max\_len** (*int*) – Max size of returned real path.

**Raises** `ssh2.exceptions.SFTPHandleError` on errors getting real path.

**Raises** `ssh2.exceptions.SFTPBufferTooSmall` on `max_len` less than real path length.

**rename** (*self*, *source\_filename*, *dest\_filename*)

Rename file.

**Parameters**

- **source\_filename** (*str*) – Old name of file.
- **dest\_filename** (*str*) – New name of file.

**rename\_ex** (*self*, *const char \*source\_filename*, *unsigned int source\_filename\_len*, *const char \*dest\_filename*, *unsigned int dest\_filename\_len*, *long flags*)

**rmdir** (*self*, *path*)

Remove directory.

**Parameters** **path** (*str*) – Directory path to remove.

**Return type** `int`

**setstat** (*self*, *path*, *SFTPAttributes attrs*)

Set file attributes.

**Parameters**

- **path** (*str*) – File path.
- **attrs** (`ssh2.sftp_handle.SFTPAttributes`) – File attributes to set.

**Return type** `int`

**stat** (*self*, *path*)

Stat file.

**Parameters** **path** (*str*) – Path of file to stat.

**Return type** `ssh2.sftp_handle.SFTPAttributes` or `LIBSSH2_ERROR_EAGAIN`

**statvfs** (*self*, *path*)

Get file system statistics from path.

**Return type** *ssh2.sftp.SFTPStatVFS* or int of error code

**symlink** (*self*, *path*, *target*)

Create symlink.

**Parameters**

- **path** (*str*) – Source file path.
- **target** (*str*) – Target file path.

**Return type** int

**unlink** (*self*, *filename*)

Delete/unlink file.

**Parameters** **filename** (*str*) – Name of file to delete/unlink.

**session**

Originating session.

## 3.5 ssh2.sftp\_handle

SFTP handle, attributes and stat VFS classes.

**class** *ssh2.sftp\_handle.SFTPAttributes*

**atime**

**filesize**

**flags**

**gid**

**mtime**

**permissions**

**uid**

**class** *ssh2.sftp\_handle.SFTPHandle*

**close** (*self*)

Close handle. Called automatically when object is deleted and/or garbage collected.

**Return type** int

**fsetstat** (*self*, *SFTPAttributes attrs*)

Set file handle attributes.

**Parameters** **attrs** (*ssh2.sftp.SFTPAttributes*) – Attributes to set.

**fstat** (*self*)

Get file stat attributes from handle.

**Return type** tuple(int, *ssh2.sftp.SFTPAttributes*)

**fstat\_ex** (*self*, *SFTPAttributes attrs*, *int setstat*)

Get or set file attributes. Clients would typically use one of the fstat or fsetstat functions instead

**fstatvfs** (*self*)

Get file system statistics for handle

**Return type** *ssh2.sftp.SFTPStatVFS*

**fsync** (*self*)

Sync file handle data.

Available from libssh2 >= 1.4.4

**Return type** *int*

**read** (*self*, *size\_t* *buffer\_maxlen=c\_ssh2.LIBSSH2\_CHANNEL\_WINDOW\_DEFAULT*)

Read buffer from file handle.

**Parameters** **buffer\_maxlen** (*int*) – Max length of buffer to return.

**Return type** *bytes*

**readdir** (*self*, *size\_t* *buffer\_maxlen=1024*)

Get directory listing from file handle, if any.

This function is a generator and should be iterated on.

File handle *must* be opened with `ssh2.sftp.SFTP.readdir()`

**Parameters** **buffer\_maxlen** – Max length of returned file entry.

**Return type** *iter(bytes)*

**readdir\_ex** (*self*, *size\_t* *longentry\_maxlen=1024*, *size\_t* *buffer\_maxlen=1024*)

Get directory listing from file handle, if any.

File handle *must* be opened with `ssh2.sftp.SFTP.readdir()`

This function is a generator and should be iterated on.

**Parameters**

- **buffer\_maxlen** – Max length of returned buffer.
- **longentry\_maxlen** – Max length of file list entry.

**Return type** *bytes*

**rewind** (*self*)

Rewind file handle to beginning of file.

**Return type** *None*

**seek** (*self*, *size\_t* *offset*)

Deprecated, use `seek64`.

Seek file to given offset.

**Parameters** **offset** (*int*) – Offset to seek to.

**Return type** *None*

**seek64** (*self*, *libssh2\_uint64\_t* *offset*)

Seek file to given 64-bit offset.

**Parameters** **offset** (*int*) – Offset to seek to.

**Return type** *None*

**tell** (*self*)

Deprecated, use tell64.

Get current file handle offset.

**Return type** `int`

**tell64** (*self*)

Get current file handle 64-bit offset.

**Return type** `int`

**write** (*self*, *bytes buf*)

Write buffer to file handle.

Returns tuple of (`error code`, `bytes written`).

In blocking mode `bytes_written` will always equal `len(buf)` if no errors have occurred which would raise exception.

In non-blocking mode `error_code` can be `LIBSSH2_ERROR_EAGAIN` and `bytes_written` can be less than `len(buf)`.

Clients should resume from that point on next call to write, ie `buf[bytes_written_in_last_call:]`.

**Parameters** `buf` (*bytes*) – Buffer to write.

**Return type** `tuple(int, int)`

**class** `ssh2.sftp_handle.SFTPStatVFS`

File system statistics

**f\_bavail**

Free blocks for non-root

**f\_bfree**

Free blocks

**f\_blocks**

Size of fs in `f_frsize` units

**f\_bsize**

File system block size

**f\_favail**

Free inodes for non-root

**f\_ffree**

Free inodes

**f\_files**

Inodes

**f\_flag**

File system mount flags.

This property is a bit mask with defined bits `LIBSSH2_SFTP_ST_RDONLY` and `LIBSSH2_SFTP_ST_NOSUID`

**f\_frsize**

Fragment size

**f\_fsid**

File system ID

**f\_namemax**  
Maximum filename length

## 3.6 ssh2.pkey

**class** `ssh2.pkey.PublicKey`  
Extension class for representing public key data from libssh2.  
Can be used for authentication via `ssh2.agent.Agent.userauth()`

**blob**  
Blob of public key data.  
**Return type** `bytes`

**blob\_len**  
Blob length of public key.  
**Return type** `int`

**comment**  
Public key comment  
**Return type** `bytes`

**magic**  
Magic number of public key.  
**Return type** `int`

## 3.7 ssh2.listener

**class** `ssh2.listener.Listener`

**forward\_accept** (*self*)

**forward\_cancel** (*self*)

## 3.8 ssh2.knownhost

**class** `ssh2.knownhost.KnownHost`  
Manage known host entries.

**add** (*self*, *bytes host*, *bytes salt*, *bytes key*, *int typemask*)  
Deprecated - use `self.addc`

**addc** (*self*, *bytes host*, *bytes key*, *int typemask*, *bytes salt=None*, *bytes comment=None*)  
Adds a host and its key to known hosts collection.

Note - `libssh2` expects correct use of hashed hosts when `LIBSSH2_KNOWNHOST_TYPE_SHA1` is part of `typemask`. Incorrect use of hashed host `typemask` without appropriate hashed host and salt values will result in host entries being added to the collection without a host name.

**Parameters**

- **host** (*bytes*) – Host to add key for.

- **key** (*bytes*) – Key to add.
- **typemask** – Bitmask of one of each from `ssh2.knownhost.LIBSSH2_KNOWNHOST_TYPE_*`, `ssh2.knownhost.LIBSSH2_KNOWNHOST_KEYENC_*` and `ssh2.knownhost.LIBSSH2_KNOWNHOST_KEY_*` for example for plain text host, raw key encoding and SSH RSA key type would be `LIBSSH2_KNOWNHOST_TYPE_PLAIN | LIBSSH2_KNOWNHOST_KEYENC_RAW | LIBSSH2_KNOWNHOST_KEY_SSHRSA`.
- **salt** (*bytes*) – Salt used for host hashing if host is hashed. Defaults to None.
- **comment** (*bytes*) – Comment to add for host. Defaults to None.

**Raises** `ssh2.exceptions.KnownHostAddError` on errors adding known host entry.

**check** (*self, bytes host, bytes key, int typemask*)

Deprecated - use `self.checkp`

**checkp** (*self, bytes host, int port, bytes key, int typemask*)

Check a host and its key against the known hosts collection and return known host entry, if any.

Note that server key provided to this function must be base64 encoded only if checking against a `self.addc` added known public key. When using `self.readfile` and a `known_hosts` file, encoding is not needed.

`ssh2.exceptions.KnownHostCheckError` is base class for all host check error exceptions and can be used to catch all host check errors.

#### Parameters

- **host** (*bytes*) – Host to check.
- **key** (*bytes*) – Key of host to check.
- **typemask** – Bitmask of one of each from `ssh2.knownhost.LIBSSH2_KNOWNHOST_TYPE_*`, `ssh2.knownhost.LIBSSH2_KNOWNHOST_KEYENC_*` and `ssh2.knownhost.LIBSSH2_KNOWNHOST_KEY_*` for example for plain text host, raw key encoding and SSH RSA key type would be `LIBSSH2_KNOWNHOST_TYPE_PLAIN | LIBSSH2_KNOWNHOST_KEYENC_RAW | LIBSSH2_KNOWNHOST_KEY_SSHRSA`.

**Raises** `ssh2.exceptions.KnownHostCheckMismatchError` on provided key mismatch error with found key from known hosts.

**Raises** `ssh2.exceptions.KnownHostCheckNotFoundError` on host not found in known hosts.

**Raises** `ssh2.exceptions.KnownHostCheckFailure` on failure checking known host entry.

**Raises** `ssh2.exceptions.KnownHostCheckError` on unknown errors checking known host.

**Return type** `ssh2.knownhost.KnownHostEntry`

**delete** (*self, KnownHostEntry entry*)

Delete given known host entry from collection of known hosts.

**Parameters** **entry** (`ssh2.knownhost.KnownHostEntry`) – Known host entry to delete.

**Raises** `ssh2.exceptions.KnownHostDeleteError` on errors deleting host entry.

**get** (*self, KnownHostEntry prev=None*)

Retrieve all host entries in known hosts collection.

**Parameters** `prev` – (Optional) Existing known host entry to start retrieval from. All hosts are retrieved when `prev` is `None` which is the default.

**Raises** `ssh2.exceptions.KnownHostGetError` on errors retrieving known host collection.

**Return type** `list(ssh2.knownhost.KnownHostEntry)`

**readfile** (*self*, *filename*, *int f\_type=c\_ssh2.LIBSSH2\_KNOWNHOST\_FILE\_OPENSSSH*)

Read known hosts file and add hosts to known hosts collection. Only OpenSSH known hosts file format is currently supported.

Returns number of successfully read host entries.

**Parameters** `filename` (*str*) – File name to read.

**Raises** `ssh2.exceptions.KnownHostReadFileError` on errors reading file.

**Return type** `int`

**readline** (*self*, *bytes line*, *int f\_type=c\_ssh2.LIBSSH2\_KNOWNHOST\_FILE\_OPENSSSH*)

Read line from known hosts file and add to known hosts collection. Only OpenSSH known hosts file format is currently supported.

Note - When using `readline`, the key values returned by `self.get` will need to be base64 encoded as `libssh2`'s `readline` does not encode them when adding, unlike `self.readfile` and `self.addc`.

**Parameters** `line` (*bytes*) – Byte string representing line to read.

**Raises** `ssh2.exceptions.KnownHostReadLineError` on errors reading line.

**writefile** (*self*, *filename*, *int f\_type=c\_ssh2.LIBSSH2\_KNOWNHOST\_FILE\_OPENSSSH*)

Write all known host entries to file. Only OpenSSH known hosts file format is currently supported.

**Parameters** `filename` (*str*) – File name to write known hosts to.

**Raises** `ssh2.exceptions.KnownHostWriteFileError` on errors writing to file.

**writeline** (*self*, *KnownHostEntry entry*, *int f\_type=c\_ssh2.LIBSSH2\_KNOWNHOST\_FILE\_OPENSSSH*, *size\_t buf\_len=1024*)

Convert a single known host entry to a single line of output for writing. Only OpenSSH known hosts file format is currently supported.

**Parameters** `entry` (*ssh2.knownhost.KnownHostEntry*) – Known host entry to write line for.

**Raises** `ssh2.exceptions.KnownHostWriteLineError` on errors writing line.

**Return type** `bytes`

**class** `ssh2.knownhost.KnownHostEntry`

Class representing a single known host entry.

**key**

Key byte string.

Key is stored base64 encoded according to `libssh2` documentation and is returned by this property as a base64 decoded byte string.

Note that in some cases, like keys added by `ssh2.knownhost.KnownHost.readline()`, the stored key is not base64 encoded, contrary to documentation, and `KnownHostEntry.key` will need to be re-encoded as base64 to get actual key.

**magic**

Entry magic number.



**name**  
Name of host.

**typemask**  
Type mask of host entry.

## 3.9 Exceptions

**exception** `ssh2.exceptions.AgentAuthenticationError`  
Bases: `ssh2.exceptions.AuthenticationError`

Raised on SSH Agent authentication errors

**exception** `ssh2.exceptions.AgentConnectionError`  
Bases: `ssh2.exceptions.AgentError`

Raised on SSH Agent connection errors

**exception** `ssh2.exceptions.AgentError`  
Bases: `ssh2.exceptions.SSH2Error`

Base class for all SSH Agent errors

**exception** `ssh2.exceptions.AgentGetIdentityError`  
Bases: `ssh2.exceptions.AgentError`

Raised on SSH Agent get identity errors

**exception** `ssh2.exceptions.AgentListIdentitiesError`  
Bases: `ssh2.exceptions.AgentError`

Raised on SSH Agent list identities errors

**exception** `ssh2.exceptions.AgentProtocolError`  
Bases: `ssh2.exceptions.SSH2Error`

Raised on SSH agent protocol errors

**exception** `ssh2.exceptions.AuthenticationError`  
Bases: `ssh2.exceptions.SSH2Error`

Base class for all authentication errors

**exception** `ssh2.exceptions.BadSocketError`  
Bases: `ssh2.exceptions.SSH2Error`

Raised on use of bad socket errors

**exception** `ssh2.exceptions.BadUseError`  
Bases: `ssh2.exceptions.SSH2Error`

Raised on API bad use errors

**exception** `ssh2.exceptions.BannerRecvError`  
Bases: `ssh2.exceptions.SessionError`

Raised on errors receiving banner

**exception** `ssh2.exceptions.BannerSendError`  
Bases: `ssh2.exceptions.SessionError`

Raised on errors sending banner

**exception** `ssh2.exceptions.BufferTooSmallError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on buffer too small errors

**exception** `ssh2.exceptions.ChannelClosedError`

Bases: `ssh2.exceptions.ChannelError`

Raised on channel closed errors

**exception** `ssh2.exceptions.ChannelEOFSentError`

Bases: `ssh2.exceptions.ChannelError`

Raised on channel EOF errors

**exception** `ssh2.exceptions.ChannelError`

Bases: `ssh2.exceptions.SSH2Error`

Base class for all channel errors

**exception** `ssh2.exceptions.ChannelFailure`

Bases: `ssh2.exceptions.ChannelError`

Raised on channel failures

**exception** `ssh2.exceptions.ChannelOutOfOrderError`

Bases: `ssh2.exceptions.ChannelError`

Raised on channel commands out of order errors

**exception** `ssh2.exceptions.ChannelPacketExceeded`

Bases: `ssh2.exceptions.ChannelError`

Raised on channel max packet length exceeded errors

**exception** `ssh2.exceptions.ChannelRequestDenied`

Bases: `ssh2.exceptions.ChannelError`

Raised on channel request denied errors

**exception** `ssh2.exceptions.ChannelUnknownError`

Bases: `ssh2.exceptions.ChannelError`

Raised on unknown channel errors

**exception** `ssh2.exceptions.ChannelWindowExceeded`

Bases: `ssh2.exceptions.ChannelError`

Raised on channel window exceeded errors

**exception** `ssh2.exceptions.CompressError`

Bases: `ssh2.exceptions.SessionError`

Raised on compression errors

**exception** `ssh2.exceptions.DecryptError`

Bases: `ssh2.exceptions.SessionError`

Raised on decryption errors

**exception** `ssh2.exceptions.EncryptError`

Bases: `ssh2.exceptions.SessionError`

Raised on encryption errors

**exception** `ssh2.exceptions.FileError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on file errors

**exception** `ssh2.exceptions.HostkeyInitError`

Bases: `ssh2.exceptions.SessionError`

Raised on errors initialiasing host key

**exception** `ssh2.exceptions.HostkeySignError`

Bases: `ssh2.exceptions.SessionError`

Raised on errors signing host key

**exception** `ssh2.exceptions.InvalidPollTypeError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on invalid poll type errors

**exception** `ssh2.exceptions.InvalidRequestError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on invalid request errors

**exception** `ssh2.exceptions.KeyExchangeError`

Bases: `ssh2.exceptions.SessionError`

Raised on errors exchanging keys

**exception** `ssh2.exceptions.KnownHostAddError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on errors adding known host entries

**exception** `ssh2.exceptions.KnownHostCheckError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on any known host check errors

**exception** `ssh2.exceptions.KnownHostCheckFailure`

Bases: `ssh2.exceptions.KnownHostCheckError`

Raised on something preventing known host check to be made

**exception** `ssh2.exceptions.KnownHostCheckMismatchError`

Bases: `ssh2.exceptions.KnownHostCheckError`

Raised on keys do not match for known host

**exception** `ssh2.exceptions.KnownHostCheckNotFoundError`

Bases: `ssh2.exceptions.KnownHostCheckError`

Raised on no match for known host check

**exception** `ssh2.exceptions.KnownHostDeleteError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on errors deleting known host entry

**exception** `ssh2.exceptions.KnownHostError`

Bases: `ssh2.exceptions.SSH2Error`

Base class for KnownHost errors

**exception** `ssh2.exceptions.KnownHostGetError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on errors retrieving known host entries

**exception** `ssh2.exceptions.KnownHostReadFileError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on errors reading from known hosts file

**exception** `ssh2.exceptions.KnownHostReadLineError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on errors reading line from known hosts file

**exception** `ssh2.exceptions.KnownHostWriteFileError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on errors writing to known hosts file

**exception** `ssh2.exceptions.KnownHostWriteLineError`

Bases: `ssh2.exceptions.KnownHostError`

Raised on errors writing line to known hosts file

**exception** `ssh2.exceptions.MethodNoneError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on invalid method errors

**exception** `ssh2.exceptions.MethodNotSupported`

Bases: `ssh2.exceptions.SessionError`

Raised on authentication method not supported errors

**exception** `ssh2.exceptions.OutOfBoundaryError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on out of boundary errors

**exception** `ssh2.exceptions.PasswordExpiredError`

Bases: `ssh2.exceptions.AuthenticationError`

Raised on password expired errors

**exception** `ssh2.exceptions.ProtocolError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on protocol errors

**exception** `ssh2.exceptions.PublicKeyError`

Bases: `ssh2.exceptions.SSH2Error`

Base class for all public key protocol errors

**exception** `ssh2.exceptions.PublicKeyInitError`

Bases: `ssh2.exceptions.PublicKeyError`

Raised on errors initialising public key system

**exception** `ssh2.exceptions.PublicKeyProtocolError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on public key protocol errors

**exception** `ssh2.exceptions.PublickeyUnverifiedError`

Bases: `ssh2.exceptions.AuthenticationError`

Raised on public key verification errors

**exception** `ssh2.exceptions.RequestDeniedError`

Bases: `ssh2.exceptions.SessionError`

Raised on request denied errors

**exception** `ssh2.exceptions.SCPProtocolError`

Bases: `ssh2.exceptions.SessionError`

Raised on SCP protocol errors

**exception** `ssh2.exceptions.SFTPError`

Bases: `ssh2.exceptions.SSH2Error`

Base class for SFTP errors

**exception** `ssh2.exceptions.SFTPHandleError`

Bases: `ssh2.exceptions.SFTPError`

Raised on SFTP handle errors

**exception** `ssh2.exceptions.SFTPProtocolError`

Bases: `ssh2.exceptions.SFTPError`

Raised on SFTP protocol errors

**exception** `ssh2.exceptions.SSH2Error`

Bases: `Exception`

Base class for all ssh2-python errors

**exception** `ssh2.exceptions.SessionError`

Bases: `ssh2.exceptions.SSH2Error`

Base class for all session errors

**exception** `ssh2.exceptions.SessionHandshakeError`

Bases: `ssh2.exceptions.SessionError`

Raised on session handshake errors

**exception** `ssh2.exceptions.SessionHostKeyError`

Bases: `ssh2.exceptions.SessionError`

Raised on errors getting server host key

**exception** `ssh2.exceptions.SessionStartupError`

Bases: `ssh2.exceptions.SessionError`

Raised on session startup errors

**exception** `ssh2.exceptions.SocketDisconnectError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on socket disconnection errors

**exception** `ssh2.exceptions.SocketRecvError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on socket receive errors

**exception** `ssh2.exceptions.SocketSendError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on socket send errors

**exception** `ssh2.exceptions.SocketTimeout`

Bases: `ssh2.exceptions.SessionError`

Raised on socket timeouts

**exception** `ssh2.exceptions.Timeout`

Bases: `ssh2.exceptions.SessionError`

Raised on timeouts

**exception** `ssh2.exceptions.UnknownError`

Bases: `ssh2.exceptions.SSH2Error`

Raised on non-specific or unknown errors

**exception** `ssh2.exceptions.ZlibError`

Bases: `ssh2.exceptions.SessionError`

Raised on zlib errors

## 3.10 Stat Info

**class** `ssh2.statinfo.StatInfo`

Representation of stat structure - libssh2 <1.7 version

`st_atime`

`st_blksize`

`st_blocks`

`st_ctime`

`st_gid`

`st_ino`

`st_mode`

`st_mtime`

`st_nlink`

`st_rdev`

`st_size`

`st_uid`

## 3.11 File Info

Available only when built on libssh2 >= 1.7

```
class ssh2.fileinfo.FileInfo
    Representation of stat structure - libssh2 >= 1.7

    st_atime
    st_blksize
    st_blocks
    st_ctime
    st_gid
    st_ino
    st_mode
    st_mtime
    st_nlink
    st_rdev
    st_size
    st_uid
```

## 3.12 Utility Functions

`ssh2.utils.handle_error_codes (int errcode) → int`

Raise appropriate exception for given error code.

Returns 0 on no error and LIBSSH2\_ERROR\_EAGAIN on EAGAIN.

**Raises** Appropriate exception from `ssh2.exceptions`.

**Parameters** `errcode` – Error code as returned by `ssh2.session.Session.last_errno()`

`ssh2.utils.ssh2_exit ()`

Call libssh2\_exit

`ssh2.utils.version (int required_version=0)`

Get libssh2 version string.

Passing in a non-zero `required_version` causes the function to return `None` if version is less than `required_version`

**Parameters** `required_version (int)` – Minimum required version

`ssh2.utils.wait_socket (_socket, Session session, timeout=1)`

Helper function for testing non-blocking mode.

This function blocks the calling thread for <timeout> seconds - to be used only for testing purposes.





## CHANGE LOG

### 4.1 0.18.0

#### 4.1.1 Changes

- Session object de-allocation no longer calls session disconnect.
- Channel object de-allocation no longer calls channel close.
- Rebuilt sources with Cython 0.29.6.
- Updated Linux and Windows binary wheels to OpenSSL 1.1.
- Updated embedded `libssh2` to latest master.
- Added Ed25519 publickey support via `libssh2` and OpenSSL upgrades.

#### 4.1.2 Packaging

- Source distribution builds would not include embedded `libssh2` module in package - #51
- Removed OSX 10.10 binary wheel builds - deprecated by Travis-CI.
- Updated embedded OpenSSL version for Windows wheel builds.

### 4.2 0.17.0.post2

#### 4.2.1 Packaging

- Updated embedded OpenSSL version for Windows wheel builds.

## 4.3 0.17.0.post1

### 4.3.1 Packaging

- Source distribution builds would not include embedded libssh2 module in package - #51
- Removed OSX 10.10 binary wheel builds - deprecated by Travis-CI.

## 4.4 0.17.0

### 4.4.1 Changes

- `SFTPHandle.write` function changed to return tuple of `return_code`, `bytes_written` for non-blocking applications to be able to handle partial writes within an SFTP write resulting from a blocked socket.
- `Channel.write*` functions changed to return tuple of `return_code`, `bytes_written` as above.

Behaviour in blocking mode has not changed. Non-blocking applications will now need to handle these functions returning a tuple and resume writes from last written offset of given data.

## 4.5 0.16.0

### 4.5.1 Changes

- Added `Session.sock` public attribute for getting socket used by `Session`.
- Source distribution default `libssh2` build target updated to upstream `libssh2` master branch.
- Added bundled `libssh2` source code for current master branch to repository and source distribution.
- Added automatic build of bundled `libssh2` code for source builds and `SYSTEM_LIBSSH2` environment variable to control building and linking against system provided `libssh2`. This will require additional steps for Windows platforms and older `libssh2` versions - see documentation.
- Updated binary wheels for all platforms to latest `libssh2`.
- Added keep alive API implementation - #47.

## 4.6 0.15.0

### 4.6.1 Changes

- Updated `session.userauth_publickey*` functions to make providing public key and private key passphrase optional.
- SFTP write calls write on all parts of buffer before returning.

## 4.6.2 Fixes

- `session.last_error()` would always return empty string.

## 4.7 0.14.0

### 4.7.1 Changes

- SFTP, SFTPHandle, Listener and PublicKeySystem functions updated to raise specific exceptions for all known libssh2 errors.
- Removed exceptions SFTPHandleError, SFTPBufferTooSmall and SFTPIOError that do not have corresponding libssh2 error codes.
- Re-generated all C code with latest Cython release.

### 4.7.2 Fixes

- Removed duplicate libssh2 definitions.
- Re-enabled system package releases.
- System package builds would not work correctly - #25.

## 4.8 0.13.0

### 4.8.1 Changes

- Upgrade embedded libssh2 in binary wheels to latest version plus enhancements.
- Adds support for ECDSA host and client keys.
- Adds support for SHA-256 host key fingerprints.
- Added SSH agent forwarding implementation.
- Windows wheels switched to OpenSSL back end.
- Windows wheels include zlib and have compression enabled.
- Windows wheels no MAC and no encryption options enabled, same as posix wheels.
- SCP functions now raise appropriate exception for all known libssh2 error codes.
- `ssh2.session.Session.disconnect` now returns 0 on success and raises exceptions on errors.
- All session `userauth_*` functions now raise specific exceptions.

## 4.8.2 Fixes

- SCP functions could not be used in non-blocking mode.

Note - libssh2 changes apply to binary wheels only. For building from source [see documentation](#).

## 4.9 0.11.0

### 4.9.1 Changes

- Session functions now raise exceptions.
- Channel functions now raise specific exceptions.
- SCP errors now raise exceptions.
- SFTP open handle errors now raise exceptions.
- Added exceptions for all known libssh2 error codes.
- Added `ssh2.utils.handle_error_codes` function for raising appropriate exception from error code.
- Added file types to `ssh2.sftp`.

### 4.9.2 Fixes

- Double de-allocation crash on objects being garbage collected in some rare cases.

## 4.10 0.10.0

### 4.10.1 Changes

- Added `ssh2.channel.Channel.shell` for opening interactive shells.

### 4.10.2 Fixes

- `ssh2.channel.Channel.process_startup` would not handle request types with no message correctly.

## 4.11 0.9.1

### 4.11.1 Fixes

- Binary wheels would have bad version info and require *git* for installation - #17

## 4.12 0.9.0

### 4.12.1 Changes

- Enabled embedded libssh2 library functionality for versions  $\geq$  1.6.0.

## 4.13 0.8.0

### 4.13.1 Changes

- Implemented known host API, all functions.
- Added *hostkey* method on *Session* class for retrieving server host key.
- Added server host key verification from known hosts file example.
- Added exceptions for all known host API errors.

## 4.14 0.7.0

### 4.14.1 Changes

- Exceptions moved from C-API to Python module

### 4.14.2 Fixes

- PyPy build support

## 4.15 0.6.0

### 4.15.1 Changes

- Implemented *last\_errno* and *set\_last\_error* session functions
- Agent authentication errors raise exceptions
- C-API refactor
- SFTP IO errors raise exceptions

## 4.15.2 Fixes

- Crash on de-allocation of channel in certain cases
- SFTP `readdir_ex` directory listing (long entry) was not returned correctly

## 4.16 0.5.5

### 4.16.1 Changes

- Accept both bytes and unicode parameters in authentication with public key from memory.

### 4.16.2 Fixes

- Unicode -> bytes parameter conversion would fail in some cases.

## 4.17 0.5.4

### 4.17.1 Fixes

- Agent authentication thread safety.

## 4.18 0.5.3

### 4.18.1 Changes

- Win32 build compatibility.
- Binary wheels for Linux, OSX and Windows, all Python versions, with embedded libssh2 and OpenSSL (embedded OpenSSL is Linux and OSX only).
- OSX CI builds.

### 4.18.2 Fixes

- Session initialisation thread safety.
- Agent thread safety.

## 4.19 0.5.2

No code changes.

## 4.20 0.5.1

### 4.20.1 Changes

- Implemented public key subsystem for public key management on remote servers
- Added all libssh2 error codes to `ssh2.error_codes`

## 4.21 0.5.0

### 4.21.1 Changes

- Implemented SFTP `statvfs` and `SFTP handle fstatvfs` methods.
- Implemented `SFTPStatVFS` extension class for file system statistics.
- SFTP `read` and `readdir` functions now return `size/error code` along with data.
- SFTP `handle fstat` now returns attributes.
- Implemented SFTP `handle readdir*` methods as python generators.
- `Block directions` function renamed to match libssh2.
- Example scripts.
- All session authentication methods now raise `AuthenticationError` on failure.

### 4.21.2 Fixes

- SFTP `readdir` functions can now be used in non-blocking mode
- Use of SFTP `opendir` via context manager

## 4.22 0.4.0

### 4.22.1 Changes

- Implemented SCP `send` and `recv` methods, all versions.
- Conditional compilation of features requiring newer versions of libssh2.
- Implemented channel `receive window adjust`, `x11_*`, `poll` and `handle extended data` methods.
- Implemented session `get/set blocking`, `get/set timeout`.
- Updated agent connection error exception name.
- Renamed session method name to match libssh2.

- Info extension classes for SCP file stat structure.

## 4.23 0.3.1

### 4.23.1 Changes

- Added context manager to SFTP handle
- Implemented SFTP write, seek, stat, fstat and last\_error methods.
- Implemented SFTPAttribute object creation and de-allocation - added unit test.

## 4.24 0.3.0

### 4.24.1 Changes

- Updated API
- Updated session, channel, agent and pkey to accept any string type arguments.
- Added get\_exit\_signal implementation for channel.



## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### S

- ssh2.agent, 19
- ssh2.channel, 15
- ssh2.exceptions, 29
- ssh2.fileinfo, 35
- ssh2.knownhost, 26
- ssh2.listener, 26
- ssh2.pkey, 26
- ssh2.session, 9
- ssh2.sftp, 19
- ssh2.sftp\_handle, 23
- ssh2.statinfo, 34
- ssh2.utils, 35



## A

add () (*ssh2.knownhost.KnownHost method*), 26  
 addc () (*ssh2.knownhost.KnownHost method*), 26  
 Agent (*class in ssh2.agent*), 19  
 agent\_auth () (*ssh2.session.Session method*), 9  
 agent\_init () (*ssh2.session.Session method*), 9  
 AgentAuthenticationError, 29  
 AgentConnectionError, 29  
 AgentError, 29  
 AgentGetIdentityError, 29  
 AgentListIdentitiesError, 29  
 AgentProtocolError, 29  
 atime (*ssh2.sftp\_handle.SFTPAttributes attribute*), 23  
 AuthenticationError, 29

## B

BadSocketError, 29  
 BadUseError, 29  
 BannerRecvError, 29  
 BannerSendError, 29  
 blob (*ssh2.pkey.PublicKey attribute*), 26  
 blob\_len (*ssh2.pkey.PublicKey attribute*), 26  
 block\_directions () (*ssh2.session.Session method*), 9  
 BufferTooSmallError, 29

## C

Channel (*class in ssh2.channel*), 15  
 ChannelClosedError, 30  
 ChannelEOFSentError, 30  
 ChannelError, 30  
 ChannelFailure, 30  
 ChannelOutOfOrderError, 30  
 ChannelPacketExceeded, 30  
 ChannelRequestDenied, 30  
 ChannelUnknownError, 30  
 ChannelWindowExceeded, 30  
 check () (*ssh2.knownhost.KnownHost method*), 27  
 checkp () (*ssh2.knownhost.KnownHost method*), 27  
 close () (*ssh2.channel.Channel method*), 15  
 close () (*ssh2.sftp\_handle.SFTPHandle method*), 23  
 comment (*ssh2.pkey.PublicKey attribute*), 26

CompressError, 30  
 connect () (*ssh2.agent.Agent method*), 19

## D

DecryptError, 30  
 delete () (*ssh2.knownhost.KnownHost method*), 27  
 direct\_tcpip () (*ssh2.session.Session method*), 10  
 direct\_tcpip\_ex () (*ssh2.session.Session method*), 10  
 disconnect () (*ssh2.agent.Agent method*), 19  
 disconnect () (*ssh2.session.Session method*), 10

## E

EncryptError, 30  
 eof () (*ssh2.channel.Channel method*), 15  
 execute () (*ssh2.channel.Channel method*), 15

## F

f\_bavail (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_bfree (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_blocks (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_bsize (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_favail (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_ffree (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_files (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_flag (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_frsize (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_fsid (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 f\_namemax (*ssh2.sftp\_handle.SFTPStatVFS attribute*), 25  
 FileError, 30  
 FileInfo (*class in ssh2.fileinfo*), 35  
 filesize (*ssh2.sftp\_handle.SFTPAttributes attribute*), 23  
 flag () (*ssh2.session.Session method*), 10  
 flags (*ssh2.sftp\_handle.SFTPAttributes attribute*), 23  
 flush () (*ssh2.channel.Channel method*), 15  
 flush\_ex () (*ssh2.channel.Channel method*), 15

`flush_stderr()` (*ssh2.channel.Channel method*), 15  
`forward_accept()` (*ssh2.listener.Listener method*), 26  
`forward_cancel()` (*ssh2.listener.Listener method*), 26  
`forward_listen()` (*ssh2.session.Session method*), 10  
`forward_listen_ex()` (*ssh2.session.Session method*), 10  
`fsetstat()` (*ssh2.sftp\_handle.SFTPHandle method*), 23  
`fstat()` (*ssh2.sftp\_handle.SFTPHandle method*), 23  
`fstat_ex()` (*ssh2.sftp\_handle.SFTPHandle method*), 23  
`fstatvfs()` (*ssh2.sftp\_handle.SFTPHandle method*), 23  
`fsync()` (*ssh2.sftp\_handle.SFTPHandle method*), 24

## G

`get()` (*ssh2.knownhost.KnownHost method*), 27  
`get_blocking()` (*ssh2.session.Session method*), 10  
`get_channel()` (*ssh2.sftp.SFTP method*), 21  
`get_exit_signal()` (*ssh2.channel.Channel method*), 15  
`get_exit_status()` (*ssh2.channel.Channel method*), 15  
`get_identities()` (*ssh2.agent.Agent method*), 19  
`get_timeout()` (*ssh2.session.Session method*), 10  
`gid` (*ssh2.sftp\_handle.SFTPAttributes attribute*), 23

## H

`handle_error_codes()` (*in module ssh2.utils*), 35  
`handle_extended_data()` (*ssh2.channel.Channel method*), 16  
`handle_extended_data2()` (*ssh2.channel.Channel method*), 16  
`handshake()` (*ssh2.session.Session method*), 10  
`hostkey()` (*ssh2.session.Session method*), 10  
`hostkey_hash()` (*ssh2.session.Session method*), 11  
`HostkeyInitError`, 31  
`HostkeySignError`, 31

## I

`ignore_extended_data()` (*ssh2.channel.Channel method*), 16  
`InvalidPollTypeError`, 31  
`InvalidRequestError`, 31

## K

`keepalive_config()` (*ssh2.session.Session method*), 11  
`keepalive_send()` (*ssh2.session.Session method*), 11  
`key` (*ssh2.knownhost.KnownHostEntry attribute*), 28

`KeyExchangeError`, 31  
`KnownHost` (*class in ssh2.knownhost*), 26  
`knownhost_init()` (*ssh2.session.Session method*), 11  
`KnownHostAddError`, 31  
`KnownHostCheckError`, 31  
`KnownHostCheckFailure`, 31  
`KnownHostCheckMismatchError`, 31  
`KnownHostCheckNotFoundError`, 31  
`KnownHostDeleteError`, 31  
`KnownHostException` (*class in ssh2.knownhost*), 28  
`KnownHostError`, 31  
`KnownHostGetError`, 31  
`KnownHostReadFileError`, 32  
`KnownHostReadLineError`, 32  
`KnownHostWriteFileError`, 32  
`KnownHostWriteLineError`, 32

## L

`last_errno()` (*ssh2.session.Session method*), 11  
`last_error()` (*ssh2.session.Session method*), 11  
`last_error()` (*ssh2.sftp.SFTP method*), 21  
`list_identities()` (*ssh2.agent.Agent method*), 19  
`Listener` (*class in ssh2.listener*), 26  
`lstat()` (*ssh2.sftp.SFTP method*), 21

## M

`magic` (*ssh2.knownhost.KnownHostException attribute*), 28  
`magic` (*ssh2.pkey.PublicKey attribute*), 26  
`method_pref()` (*ssh2.session.Session method*), 11  
`MethodNoneError`, 32  
`MethodNotSupported`, 32  
`methods()` (*ssh2.session.Session method*), 12  
`mkdir()` (*ssh2.sftp.SFTP method*), 21  
`module`

- `ssh2.agent`, 19
- `ssh2.channel`, 15
- `ssh2.exceptions`, 29
- `ssh2.fileinfo`, 35
- `ssh2.knownhost`, 26
- `ssh2.listener`, 26
- `ssh2.pkey`, 26
- `ssh2.session`, 9
- `ssh2.sftp`, 19
- `ssh2.sftp_handle`, 23
- `ssh2.statinfo`, 34
- `ssh2.utils`, 35

`mtime` (*ssh2.sftp\_handle.SFTPAttributes attribute*), 23

## N

`name` (*ssh2.knownhost.KnownHostException attribute*), 28

## O

`open()` (*ssh2.sftp.SFTP method*), 21

open\_ex() (*ssh2.sftp.SFTP method*), 22  
 open\_session() (*ssh2.session.Session method*), 12  
 opendir() (*ssh2.sftp.SFTP method*), 22  
 OutOfBoundaryError, 32

## P

PasswordExpiredError, 32  
 permissions (*ssh2.sftp\_handle.SFTPAttributes attribute*), 23  
 poll\_channel\_read() (*ssh2.channel.Channel method*), 16  
 process\_startup() (*ssh2.channel.Channel method*), 16  
 ProtocolError, 32  
 pty() (*ssh2.channel.Channel method*), 16  
 PublicKey (*class in ssh2.pkey*), 26  
 publickey\_init() (*ssh2.session.Session method*), 13  
 PublicKeyError, 32  
 PublicKeyInitError, 32  
 PublicKeyProtocolError, 32  
 PublickeyUnverifiedError, 32

## R

read() (*ssh2.channel.Channel method*), 16  
 read() (*ssh2.sftp\_handle.SFTPHandle method*), 24  
 read\_ex() (*ssh2.channel.Channel method*), 16  
 read\_stderr() (*ssh2.channel.Channel method*), 16  
 readdir() (*ssh2.sftp\_handle.SFTPHandle method*), 24  
 readdir\_ex() (*ssh2.sftp\_handle.SFTPHandle method*), 24  
 readfile() (*ssh2.knownhost.KnownHost method*), 28  
 readline() (*ssh2.knownhost.KnownHost method*), 28  
 realpath() (*ssh2.sftp.SFTP method*), 22  
 receive\_window\_adjust() (*ssh2.channel.Channel method*), 16  
 receive\_window\_adjust2() (*ssh2.channel.Channel method*), 16  
 rename() (*ssh2.sftp.SFTP method*), 22  
 rename\_ex() (*ssh2.sftp.SFTP method*), 22  
 RequestDeniedError, 33  
 rewind() (*ssh2.sftp\_handle.SFTPHandle method*), 24  
 rmdir() (*ssh2.sftp.SFTP method*), 22

## S

scp\_recv() (*ssh2.session.Session method*), 13  
 scp\_recv2() (*ssh2.session.Session method*), 13  
 scp\_send() (*ssh2.session.Session method*), 13  
 scp\_send64() (*ssh2.session.Session method*), 13  
 SCPProtocolError, 33  
 seek() (*ssh2.sftp\_handle.SFTPHandle method*), 24  
 seek64() (*ssh2.sftp\_handle.SFTPHandle method*), 24  
 send\_eof() (*ssh2.channel.Channel method*), 16

Session (*class in ssh2.session*), 9  
 session (*ssh2.channel.Channel attribute*), 19  
 session (*ssh2.sftp.SFTP attribute*), 23  
 SessionError, 33  
 SessionHandshakeError, 33  
 SessionHostKeyError, 33  
 SessionStartupError, 33  
 set\_blocking() (*ssh2.session.Session method*), 13  
 set\_last\_error() (*ssh2.session.Session method*), 13  
 set\_timeout() (*ssh2.session.Session method*), 13  
 setenv() (*ssh2.channel.Channel method*), 17  
 setstat() (*ssh2.sftp.SFTP method*), 22  
 SFTP (*class in ssh2.sftp*), 21  
 sftp\_init() (*ssh2.session.Session method*), 13  
 SFTPAttributes (*class in ssh2.sftp\_handle*), 23  
 SFTPErrors, 33  
 SFTPHandle (*class in ssh2.sftp\_handle*), 23  
 SFTPHandleError, 33  
 SFTPProtocolError, 33  
 SFTPStatVFS (*class in ssh2.sftp\_handle*), 25  
 shell() (*ssh2.channel.Channel method*), 17  
 sock (*ssh2.session.Session attribute*), 15  
 SocketDisconnectError, 33  
 SocketRecvError, 33  
 SocketSendError, 33  
 SocketTimeout, 34  
 ssh2.agent  
   module, 19  
 ssh2.channel  
   module, 15  
 ssh2.exceptions  
   module, 29  
 ssh2.fileinfo  
   module, 35  
 ssh2.knownhost  
   module, 26  
 ssh2.listener  
   module, 26  
 ssh2.pkey  
   module, 26  
 ssh2.session  
   module, 9  
 ssh2.sftp  
   module, 19  
 ssh2.sftp\_handle  
   module, 23  
 ssh2.statinfo  
   module, 34  
 ssh2.utils  
   module, 35  
 ssh2\_exit() (*in module ssh2.utils*), 35  
 SSH2Error, 33  
 st\_atime (*ssh2.fileinfo.FileInfo attribute*), 35

`st_atime` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_blksize` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_blksize` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_blocks` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_blocks` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_ctime` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_ctime` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_gid` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_gid` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_ino` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_ino` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_mode` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_mode` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_mtime` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_mtime` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_nlink` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_nlink` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_rdev` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_rdev` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_size` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_size` (*ssh2.statinfo.StatInfo* attribute), 34  
`st_uid` (*ssh2.fileinfo.FileInfo* attribute), 35  
`st_uid` (*ssh2.statinfo.StatInfo* attribute), 34  
`startup()` (*ssh2.session.Session* method), 13  
`stat()` (*ssh2.sftp.SFTP* method), 22  
`StatInfo` (class in *ssh2.statinfo*), 34  
`statvfs()` (*ssh2.sftp.SFTP* method), 22  
`subsystem()` (*ssh2.channel.Channel* method), 17  
`supported_algs()` (*ssh2.session.Session* method), 14  
`symlink()` (*ssh2.sftp.SFTP* method), 23

## T

`tell()` (*ssh2.sftp\_handle.SFTPHandle* method), 24  
`tell64()` (*ssh2.sftp\_handle.SFTPHandle* method), 25  
Timeout, 34  
`typemask` (*ssh2.knownhost.KnownHostEntry* attribute), 29

## U

`uid` (*ssh2.sftp\_handle.SFTPAttributes* attribute), 23  
UnknownError, 34  
`unlink()` (*ssh2.sftp.SFTP* method), 23  
`userauth()` (*ssh2.agent.Agent* method), 19  
`userauth_authenticated()`  
(*ssh2.session.Session* method), 14  
`userauth_hostbased_fromfile()`  
(*ssh2.session.Session* method), 14  
`userauth_keyboardinteractive()`  
(*ssh2.session.Session* method), 14  
`userauth_list()` (*ssh2.session.Session* method), 14  
`userauth_password()` (*ssh2.session.Session* method), 14

`userauth_publickey()` (*ssh2.session.Session* method), 15  
`userauth_publickey_fromfile()`  
(*ssh2.session.Session* method), 15  
`userauth_publickey_frommemory()`  
(*ssh2.session.Session* method), 15

## V

`version()` (in module *ssh2.utils*), 35

## W

`wait_closed()` (*ssh2.channel.Channel* method), 17  
`wait_eof()` (*ssh2.channel.Channel* method), 17  
`wait_socket()` (in module *ssh2.utils*), 35  
`window_read()` (*ssh2.channel.Channel* method), 17  
`window_read_ex()` (*ssh2.channel.Channel* method), 17  
`window_write()` (*ssh2.channel.Channel* method), 17  
`window_write_ex()` (*ssh2.channel.Channel* method), 17  
`write()` (*ssh2.channel.Channel* method), 17  
`write()` (*ssh2.sftp\_handle.SFTPHandle* method), 25  
`write_ex()` (*ssh2.channel.Channel* method), 18  
`write_stderr()` (*ssh2.channel.Channel* method), 18  
`writeln()` (*ssh2.knownhost.KnownHost* method), 28  
`writeline()` (*ssh2.knownhost.KnownHost* method), 28

## X

`x11_req()` (*ssh2.channel.Channel* method), 18  
`x11_req_ex()` (*ssh2.channel.Channel* method), 19

## Z

ZlibError, 34