
RedSSH Documentation

Release 3.0.0 stable

Red_M

Dec 26, 2022

CONTENTS:

1	RedSSH	1
2	RedSSH.sftp	7
3	RedSSH.scp	9
4	RedSSH.enums	11
5	RedSSH.exceptions	13
6	Indices and tables	15
	Python Module Index	17
	Index	19

REDSSH

```
class redssh.RedSSH(encoding='utf8', terminal='vt100', known_hosts=None,
                    ssh_host_key_verification=SSHHostKeyVerify.warn, ssh_keepalive_interval=0.0,
                    set_flags={}, method_preferences={}, callbacks={}, auto_terminate_tunnels=False,
                    tcp_nodelay=False)
```

Bases: `object`

Instances the start of an SSH connection. Extra options are available after `redssh.RedSSH.connect()` is called.

Parameters

encoding (str) – Set the encoding to something other than the default of 'utf8' when your target SSH server doesn't return UTF-8.

close_tunnels()

Closes all SSH tunnels if any are open.

connect(hostname, port=22, username="", password=None, allow_agent=False, host_based=None, key_filepath=None, passphrase=None, look_for_keys=False, sock=None, timeout=None)

Warning: Some authentication methods are not yet supported!

Parameters

- **hostname** (str) – Hostname to connect to.
- **port** (int) – SSH port to connect to.
- **username** (str) – Username to connect as to the remote server.
- **password** (str) – Password to offer to the remote server for authentication.
- **allow_agent** (bool) – Allow the local SSH key agent to offer the keys held in it for authentication.
- **host_based** (bool) – Allow the local SSH host keys to be used for authentication. NOT IMPLEMENTED!
- **key_filepath** (array/str) – Array of filenames to offer to the remote server. Can be a string for a single key.
- **passphrase** (str) – Passphrase to decrypt any keys offered to the remote server.
- **look_for_keys** (bool) – Enable offering keys in ~/.ssh automatically. NOT IMPLEMENTED!

- **sock** (`socket.socket()`) – A pre-connected socket to the remote server. Useful if you have strange network requirements.
- **timeout** (`float`) – Timeout for the socket connection to the remote server.

dynamic_tunnel(*local_port*, *bind_addr*='127.0.0.1', *error_level*=`TunnelErrorLevel.warn`)

Opens a SOCKS proxy AKA gateway or dynamic port the same way the `-D` option does for the OpenSSH client.

Providing a `0` for the local port will mean the OS will assign an unbound port for you. This port number will be provided to you by this function.

Parameters

- **local_port** (`int`) – The local port on the local machine to bind to.
- **bind_addr** (`str`) – The bind address on this machine to bind to for the local port.
- **error_level** (`redssh.enums.TunnelErrorLevel`) – The level of verbosity that errors in tunnel threads will use.

Returns

`int` The local port that has been bound.

eof()

Returns True or False when the main channel has recieved an EOF.

execute_command(*command*)

Run a command. This will block as the command executes.

Parameters

command (`str`) – Command to execute.

Returns

`tuple (int, str)` - of (`return_code`, `command_output`)

exit()

Kill the current session if connected.

flush()

Flush all data on the primary channel's stdin to the remote connection. Only works if connected, otherwise returns `0`.

Returns

`int` - Amount of bytes sent to remote machine.

forward_x11()

Start forwarding an X11 display.

Returns

`None`

last_error()

Get the last error from the client session.

Returns

`str`

local_tunnel(*local_port*, *remote_host*, *remote_port*, *bind_addr*='127.0.0.1',
error_level=`TunnelErrorLevel.debug`)

Forwards a port on the remote machine the same way the `-L` option does for the OpenSSH client.

Providing a `0` for the local port will mean the OS will assign an unbound port for you. This port number will be provided to you by this function.

Parameters

- **local_port** (int) – The local port on the local machine to bind to.
- **remote_host** (str) – The remote host to connect to via the remote machine.
- **remote_port** (int) – The remote host's port to connect to via the remote machine.
- **bind_addr** (str) – The bind address on this machine to bind to for the local port.
- **error_level** ([`redssh.enums.TunnelErrorLevel`](#)) – The level of verbosity that errors in tunnel threads will use.

Returns

int The local port that has been bound.

methods(*method*)

Returns what value was settled on during session negotiation.

read(*block=False*)

Recieve data from the remote session. Only works if the current session has made it past the login process.

Parameters

block (bool) – Block until data is received from the remote server. True will block until data is recieved and False may return `b''` if no data is available from the remote server.

Returns

generator - A generator of byte strings that has been recieved in the time given.

remote_tunnel(*local_port, remote_host, remote_port, bind_addr='127.0.0.1', error_level=TunnelErrorLevel.warn*)

Forwards a port to the remote machine via the local machine the same way the `-R` option does for the OpenSSH client.

Parameters

- **local_port** (int) – The local port on the remote side for clients to connect to.
- **remote_host** (str) – The remote host to connect to via the local machine.
- **remote_port** (int) – The remote host's port to connect to via the local machine.
- **error_level** ([`redssh.enums.TunnelErrorLevel`](#)) – The level of verbosity that errors in tunnel threads will use.

Returns

None

send(*string*)

Send data to the remote session. Only works if the current session has made it past the login process.

Parameters

string (str) – String to send to the remote session.

Returns

int - Amount of bytes sent to remote machine.

setenv(*varname*, *value*)

Set an environment variable on the channel.

Parameters

- **varname** (str) – Name of environment variable to set on the remote channel.
- **value** (str) – Value to set **varname** to.

Returns

None

shutdown_tunnel(*tunnel_type*, *sport*, *rhost*=None, *rport*=None, *bind_addr*='127.0.0.1')

Closes an open tunnel. Provide the same arguments to this that was given for opening the tunnel.

Examples:

local_tunnel(9999, 'localhost', 8888) would be *shutdown_tunnel*(*redssh.enums.TunnelType.local*, 9999, 'localhost', 8888)

remote_tunnel(7777, 'localhost', 8888) would be *shutdown_tunnel*(*redssh.enums.TunnelType.remote*, 7777, 'localhost', 8888)

dynamic_tunnel(9999) would be *shutdown_tunnel*(*redssh.enums.TunnelType.dynamic*, 9999)

dynamic_tunnel(9999, '10.0.0.1') would be *shutdown_tunnel*(*redssh.enums.TunnelType.dynamic*, 9999, *bind_addr*='10.0.0.1')

Parameters

- **tunnel_type** (*redssh.enums.TunnelType*) – The tunnel type to shutdown.
- **sport** (str) – The bound port for local and dynamic tunnels or the local port on the remote side for remote tunnels.
- **rhost** (str) – The remote host for local and remote tunnels.
- **rport** (int) – The remote port for local and remote tunnels.
- **bind_addr** (str) – The bind address used for local and dynamic tunnels.

Returns

None

start_scp()

Start the SCP client. If the client or server doesn't support SCP, SFTP will be started instead, this is due to SCP being deprecated.

Returns

None

start_sftp()

Start the SFTP client. The client will be available at *self.sftp* and will be an instance of *redssh.sftp.RedSFTP*

Returns

None

tunnel_is_alive(*tunnel_type*, *sport*, *rhost*=None, *rport*=None, *bind_addr*='127.0.0.1')

Checks if a tunnel is alive. Provide the same arguments to this that was given for opening the tunnel.

Examples:

local_tunnel(9999, 'localhost', 8888) would be *tunnel_is_alive*(*redssh.enums.TunnelType.local*, 9999, 'localhost', 8888)

remote_tunnel(7777, 'localhost', 8888) would be *tunnel_is_alive*(*redssh.enums.TunnelType.remote*, 7777, 'localhost', 8888)

dynamic_tunnel(9999) would be *tunnel_is_alive*(*redssh.enums.TunnelType.dynamic*, 9999)

dynamic_tunnel(9999, '10.0.0.1') would be *tunnel_is_alive*(*redssh.enums.TunnelType.dynamic*, 9999, *bind_addr*='10.0.0.1')

Parameters

- **tunnel_type** ([redssh.enums.TunnelType](#)) – The tunnel type to shutdown.
- **sport** (str) – The bound port for local and dynamic tunnels or the local port on the remote side for remote tunnels.
- **rhost** (str) – The remote host for local and remote tunnels.
- **rport** (int) – The remote port for local and remote tunnels.
- **bind_addr** (str) – The bind address used for local and dynamic tunnels.

Returns

bool, if bad tunnel type provided returns None

write(string)

See [redssh.RedSSH.send\(\)](#)

CHAPTER
TWO

REDSSH.SFTP

CHAPTER
THREE

REDSSH.SCP

REDSSH.ENUMS

class redssh.enums.SSHClient(*value*)

Bases: *StrEnum*

An enumeration.

class redssh.enums.SSHHostKeyVerify(*value*)

Bases: *IntEnum*

An enumeration.

class redssh.enums.StrEnum(*value*)

Bases: *str*, *Enum*

An enumeration.

class redssh.enums.TunnelErrorLevel(*value*)

Bases: *IntEnum*

An enumeration.

class redssh.enums.TunnelType(*value*)

Bases: *StrEnum*

An enumeration.

REDSSH.EXCEPTIONS

exception `redssh.exceptions.AuthenticationFailedException`(*authentication*)

Bases: [*RedSSHException*](#)

Failed to authenticate via supplied methods.

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `redssh.exceptions.NoAuthenticationOfferedException`

Bases: [*RedSSHException*](#)

Failed to authenticate because no methods were supplied.

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `redssh.exceptions.RedSSHException`

Bases: [*Exception*](#)

Base error class for sub classing.

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

r

`redssh.enums`, [11](#)

`redssh.exceptions`, [13](#)

INDEX

A

AuthenticationFailedException, 13

C

close_tunnels() (*redssh.RedSSH method*), 1

connect() (*redssh.RedSSH method*), 1

D

dynamic_tunnel() (*redssh.RedSSH method*), 2

E

eof() (*redssh.RedSSH method*), 2

execute_command() (*redssh.RedSSH method*), 2

exit() (*redssh.RedSSH method*), 2

F

flush() (*redssh.RedSSH method*), 2

forward_x11() (*redssh.RedSSH method*), 2

L

last_error() (*redssh.RedSSH method*), 2

local_tunnel() (*redssh.RedSSH method*), 2

M

methods() (*redssh.RedSSH method*), 3

module

 redssh.enums, 11

 redssh.exceptions, 13

N

NoAuthenticationOfferedException, 13

R

read() (*redssh.RedSSH method*), 3

RedSSH (*class in redssh*), 1

redssh.enums

 module, 11

redssh.exceptions

 module, 13

RedSSHException, 13

remote_tunnel() (*redssh.RedSSH method*), 3

S

send() (*redssh.RedSSH method*), 3

setenv() (*redssh.RedSSH method*), 3

shutdown_tunnel() (*redssh.RedSSH method*), 4

SSHClient (*class in redssh.enums*), 11

SSHHostKeyVerify (*class in redssh.enums*), 11

start_scp() (*redssh.RedSSH method*), 4

start_sftp() (*redssh.RedSSH method*), 4

StrEnum (*class in redssh.enums*), 11

T

tunnel_is_alive() (*redssh.RedSSH method*), 4

TunnelErrorLevel (*class in redssh.enums*), 11

TunnelType (*class in redssh.enums*), 11

W

with_traceback() (*redssh.exceptions.AuthenticationFailedException method*), 13

with_traceback() (*redssh.exceptions.NoAuthenticationOfferedException method*), 13

with_traceback() (*redssh.exceptions.RedSSHException method*), 13

write() (*redssh.RedSSH method*), 5