pyssh-ctypes Documentation

Release 0.2

Andrey Antukh

September 21, 2016

Contents

Is a python, object oriented wrapper for libssh, build with ctypes.

CHAPTER 1

Features

- SSH command execution with streaming api.
- SFTP subsystem with random access to remote files.
- Compatible with python3, python2 and pypy.
- Unique dependece: `libssh >= 0.5`

CHAPTER 2

How to install

For normal use, you can use a standard python distutils `setup.py` file:

```
python setup.py install
```

Or:

```
pip install pyssh-ctypes
```

CHAPTER 3

Contents:

3.1 Usage examples

3.1.1 Simple command execution

```
>>> import pyssh
>>> s = pyssh.new_session(hostname="localhost", port="22")
>>> r = s.execute("uname -a")
>>> r.as_bytes()
b'Linux vaio.niwi.be 3.5.3-1-ARCH #1 SMP PREEMPT Sun Aug 26 09:14:51 CEST 2012 x86_64 GNU/Linux\n'
>>> r.return_code
0
```

3.1.2 Random access on remote file with sftp

```
>>> import os
>>> import pyssh
>>> session = pyssh.new_session(hostname="localhost")
>>> sftp = session.create_sftp()
>>> f = sftp.open("/tmp/some-file", (os.O_RDWR | os.O_CREAT))
>>> f.tell()
0
>>> f.write(b'Hello World')
>>> f.tell()
11
>>> f.seek(0)
True
>>> f.read(5)
b'Hello'
>>> f.read()
b' World'
```

3.2 Internals reference

3.2.1 Entry point

`pyssh.new_session`(*hostname='localhost'*, *port='22'*, *username=None*, *password=None*, *passphrase=None*, *connect_on_init=False*, *verify_knownhost_callback=None*)

Shortcut method for create new session instance.

Session by default has lazy connection management. It only connects when it is need. But this this function you can pass *connect_on_init* parameter with True and session connects to the remote server before it are returned.

Parameters

- **hostname** (*str*) – remote ip or host
- **port** (*int*) – remote port
- **username** (*str*) – remote user name with which you want to authenticate
- **password** (*str*) – remote user password.
- **passphrase** (*str*) – passphrase in case you would authenticate with pubkey
- **connect_on_init** (*bool*) – determines the lazyness of connection with remote server.

`pyssh.connect`(*hostname='localhost'*, *port='22'*, *username=None*, *password=None*, *passphrase=None*)

Shortcut method for create new session and connects to remote server.

Parameters

- **hostname** (*str*) – remote ip or host
- **port** (*int*) – remote port
- **username** (*str*) – remote user name with which you want to authenticate
- **password** (*str*) – remote user password.
- **passphrase** (*str*) – passphrase in case you would authenticate with pubkey

NOTE: this method is deprecated.

3.2.2 Session

class `pyssh.session.Session`(*hostname*, *port=22*, *username=None*, *password=None*, *passphrase=None*, *verify_knownhost_callback=None*)

SSH Session wrapper.

Actually accepts two methods for authentication: the simple a simple password or a pubkey. If password is not provided, attempts using pubkey, with or without pasphrase.

Variables

- ***session*** (*pointer*) – c ssh session pointer
- ***username*** (*bytes*) – current username

Parameters

- **hostname** (*str*) – remote ip or host
- **port** (*int*) – remote port
- **username** (*str*) – remote user name with which you want to authenticate

- **password** (*str*) – remote user password.
- **passphrase** (*str*) – passphrase in case you would authenticate with pubkey
- **verify_knownhost_callback** (*func*) – function which gets called upon connecting to host. Should return True if connection is allowed, False otherwise. The only parameter to the function is remote host key SHA1 hash. WARNING: you should always verify host signature!

close()

Close initialized ssh connection.

create_sftp()

Create a new sftp session throught current ssh channel.

Returns Sftp instance

Return type *pyssh.sftp.Sftp*

create_shell(*pty_size=(80, 24)*, *env={}*)

Creates a new shell session throught current ssh channel.

Parameters

- **pty_size** (*tuple*) – in case of shell is true this indicates the size of a virtual terminal
- **env** (*dict*) – addiotional environ variables

execute(*command*, *lazy=False*)

Execute command on remote host.

This command can return *Result* or *LazyResult* depending of lazy parameter.

Parameters

- **command** (*str*) – command string
- **lazy** (*bool*) – set true for return a lazy result instead a evaluated. Useful for execute commands with large output (default: False)

Returns Result instance

Return type *pyssh.result.Result*

password = None

session = None

username = None

class pyssh.result.**LazyResult**(*session*, *command*)

Lazy command execution result wrapper.

This wrapper implements a iterator interface.

as_bytes()

Launch the command and return a result as bytes.

Returns bytes chunk of command execution result

Return type bytes

as_str()

Launch the command and return a result as unicode string

Returns unicode chunk of command execution result

Return type str/unicode

return_code

wait()

Waits a complete command execution and returns the return code

Returns execution result return code

Return type int

class pyssh.result.**Result**(**args*, ***kwargs*)

Consumed version of LazyResult. Useful for simple command execution.

as_bytes()

Return a cached result.

Returns bytes chunk of command execution result

Return type bytes

as_str()

Launch the command and return a result as unicode string

Returns unicode chunk of command execution result

Return type str/unicode

return_code

wait()

3.2.3 Shell

class pyssh.shell.**Shell**(*session*, *pty_size*, *env*)

Shell session.

channel

read(*n*)

Read bytes from remote shell.

This method always return value, if not bytes available to read it returns an empty bytestring.

Parameters **n** (*int*) – number of bytes to read

Returns bytestring of readed data.

Return type bytes

write(*data*)

Write bytes to remote shell.

The *data* parameter accept both str and bytes, if you passes str (unicode) is automatically converted to bytes using utf-8 encoding.

Parameters **data** (*bytes*) – arbitrary length of bytes.

Returns a number of bytes written to remote shell.

Return type int

3.2.4 SFTP

class `pyssh.sftp.`**`Sftp`**(*session*, *buffer_size=16384*)

Sftp wrapper.

Exposes api for interacting with sftp subsystem: put or get files, open files with random read-write access, etc.

Variables

- ***sftp*** (*ponter*) – c sftp session pointer
- ***session*** (*pointer*) – c ssh session pointer

Parameters **session** (`pyssh.session.Session`) – initialized and connected *pyssh.session.Session* instance.

get(*remote_path*, *local_path*)

Get a remote file to local.

Parameters

- **remote_path** (*str*) – remote file path
- **local_path** (*str*) – local file path

open(*path*, *mode*)

Open a remote file.

Parameters

- **path** (*str*) – remote file path
- **mode** (*int*) – open file model (see http://docs.python.org/3.3/library/os.html#open-flag-constants)

Returns SFTP File wrapper

Return type pyssh.SftpFile

put(*path*, *remote_path*)

Puts the local file to remote host.

Parameters

- **path** (*str*) – local file path
- **remote_path** (*str*) – remote file path

session = None

sftp = None

class `pyssh.sftp.`**`SftpFile`**(*path*, *mode*, *sftp_wrapper*)

SFTP File wrapper

close()

Close a opened file.

read(*num=None*, *buffer_length=1024*)

Read from remote file.

Parameters **num** (*int*) – number of bytes to read, if num is None reads all.

Returns readed bytes chunk

Return type bytes

seek(*offset*)

Change position on a remote file.

Parameters **offset** (*int*) – file position

Returns boolean value if seek is success or not

Return type bool

tell()

Query the current position on a file.

Returns a current position.

Return type int

write(*data*)

Write bytes to remote file.

Parameters **data** (*bytes*) – bytes chunk of data

Returns number of bytes are written

Return type int

CHAPTER 4

Indices and tables

- genindex
- modindex
- search