rfslib

Release 2.1.7

Přemysl Šťastný

Aug 28, 2021

CONTENTS:

1 rfslib.abstract_pconnection module 3

2 rfslib.sftp_pconnection module 9

3 rfslib.ftp_pconnection module 13

4 rfslib.smb12_pconnection module 17

5 rfslib.smb23_pconnection module 21

6 rfslib.fs_pconnection module 25

7 rfslib.path_utils module 29

Python Module Index 31

Index 33

This is a documentation of rfslib.

To create a new development enviroment, it is recommended to create python virtual enviroment and install dependencies in requirements.txt

If you want to create a new pdf documentation, you are required to install also texlive on your system.

CHAPTER

ONE

RFSLIB.ABSTRACT_PCONNECTION MODULE

class rfslib.abstract_pconnection.**PConnection**(*settings:* rfslib.abstract_pconnection.p_connection_settings)

Bases: abc.ABC

__init__(*settings:* rfslib.abstract_pconnection.p_connection_settings)

The constructor of a abstract class. If it is not called from child class, the behavior is undefined.

If local_encoding and remote_encoding have same values, no recoding is done. Analogically if local_crlf and remote_crlf is same, no substitution between LF and CRLF is done.

Parameters **settings** – A p_connection_settings object with all generic settings for PConnection. Be sure, that all needed attributes are present, or AttributeError will be raised.

abstract _exists(*remote_path: str*) → bool

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns False.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

abstract _isdir(*remote_path: str*) → bool

Protected method which checks, whether a remote file is a directory.

Parameters **remote_path** – A path of a directory.

Returns True, if remote file is folder. False, if it isn't a folder. Undefined if the file doesn't exist.

abstract _lexists(*remote_path: str*) → bool

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns True.

KNOWN BUG: Behavior is undefined in case of broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

abstract _listdir(*remote_path: str*) → List[str]

Protected method which returns a list of files in the folder including hidden files. Undefined if the remote file doesn't exist or isn't a folder.

Parameters **remote_path** – The remote path of a remote folder.

Returns List of files in the remote folder

abstract _mkdir(*remote_path: str*)

Protected method which creates a new directory. Behavior is undefined if remote folder already exist, or destination folder doesn't exist.

Parameters **remote_path** – A path of a new remote directory.

abstract _pull(*remote_path: str*, *local_path: str*)
Protected method which downloads/pulls a nondirectory file from a remote storage to a local storage in the binary form. Behavior is undefined if source file or destination folder doesn't exist.

Parameters

- **remote_path** – Path of a remote file to download.
- **local_path** – Path of a local file, where to download/pull a remote file or local file already exists.

abstract _push(*local_path: str*, *remote_path: str*)
Protected method which uploads/pushes a nondirectory file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder or source file doesn't exist, source is directory or remote file already exists.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

abstract _rename(*old_name: str*, *new_name: str*)
Protected method which renames/moves a file. Behavior is undefined, if *new_name* file exists or *old_name* file doesn't exist.

Parameters

- **old_name** – Remote path a file to move.
- **new_name** – Remote path to which move the file.

abstract _rmdir(*remote_path: str*)
Protected method which removes an empty remote directory. Behavior is undefined if remote directory doesn't exist or it isn't empty.

Parameters **remote_path** – Path of an empty remote directory to delete.

abstract _stat(*remote_path: str*) → os.stat_result
Protected method which returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file. Undefined behavior if remote file doesn't exist or it is a broken symlink.

Parameters **remote_path** – Path of a remote file.

Returns The function returns os.stat_result like object, which is further parsed by _stat_unpack function. For more details please see source code.

abstract _unlink(*remote_path: str*)
Protected method which removes a nondirectory file. Behavior is undefined if remote file doesn't exist or is a directory.

Parameters **remote_path** – Path of a remote regular file to delete.

abstract close()
Method to close the opened connection.

cp(*old_names*, *new_name*, *recursive=False*)

dcp(*old_names*, *target_dir*, *recursive=False*)

dmv(*old_names*, *target_dir*)

exists(*remote_path: str*) → bool
Method which checks, whether a remote file exist. Returns False for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

fcp(*old_name*, *new_name*)

find(*remote_path*, *child_first=False*)

fmv(*old_name*, *new_name*)

get_default_dmask() → int
Returns default_dmask settings. For moe details see p_connection_settings.

get_default_fmask() → int
Returns default_fmask settings. For more details see p_connection_settings.

get_settings() → *rfslib.abstract_pconnection.p_connection_settings*
The procedure sets all generic settings for PConnection.

Returns A p_connection_settings object with all generic settings of PConnection.

isdir(*remote_path*)

lexists(*remote_path*)
Method which checks, whether a remote file exist. Returns True for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

listdir(*remote_path*)

ls(*remote_path*)

lstat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Doesn't follow symlinks.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)
Uploads/pushes a file from a local storage to a remote storage in the binary form.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

rename(*old_name*, *new_name*)

rm(*remote_path*, *recursive=False*)

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

set_settings(*settings:* rfslib.abstract_pconnection.p_connection_settings)
The procedure sets all generic settings for PConnection.

Parameters **settings** – A p_connection_settings object with all generic settings for PConnection. If some attribute in object is missing, no operation will be done with it.

stat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

class rfslib.abstract_pconnection.**p_connection_settings**
Bases: object

This object represents settings appliable for all PConnection instances (instances of class, which inherits from PConnection).

__init__()
The constructor inicializes the class to default values.

default_dmask: int = 18
If mode (permissions) of a directory can't be fetched, this value will be used instead of it.

default_fmask: int = 91
If mode (permissions) of a nondirectory file can't be fetched, this value will be used instead of it.

direct_write: bool = False
NOT IMPLEMENTED YET. If True, push will write output directly to file. If False all push operations on regular files will create firstly tmp file in target folder and then move result to file.

local_crlf: bool = False
Does local files use CRLF? If True, it is supposed, they do. If False, it is supposed, they use LF.

local_encoding: str = 'UTF8'
The encoding of local text files. (eg. 'UTF8')

remote_crlf: bool = False
Does remote files use CRLF? If True, it is supposed, they do. If False, it is supposed, they use LF.

remote_encoding: str = 'UTF8'
The encoding of remote text files. (eg. 'cp1250')

skip_validation: bool = False
NOT IMPLEMENTED YED. If True, all validations of input will be skipped. Undefined behavior may happen if input is wrong. Increses performance.

text_transmission: bool = False
If true, all files, which will be transmitted, will be recoded from local_encoding to remote_encoding and from local_crlf to remote_crlf. If False, there will be no encoding done during transmission.

class rfslib.abstract_pconnection.**p_stat_result**
Bases: object

Representation of the attributes of a file (or proxied file). It attemps to mirror the object returned by os.stat as closely as possible.

__init__()

st_atime = None
This is the time of the last access of file data.

st_gid = None
This field contains the ID of the group owner of the file.

st_mode = None
This field contains the file type and mode.

st_mtime = None
This is the time of last modification of file data.

st_nlink = None
This field contains the number of hard links to the file.

st_size = None
This field gives the size of the file (if it is a regular file or a symbolic link) in bytes. The size of a symbolic link is the length of the pathname it contains, without a terminating null byte.

st_uid = None
This field contains the user ID of the owner of the file.

CHAPTER

TWO

RFSLIB.SFTP_PCONNECTION MODULE

class rfslib.sftp_pconnection.**SftpPConnection**(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *username: str*, *password: Optional[str] = None*, *keyfile: str = '~/.ssh/id_rsa'*, *port: int = 22*, *no_host_key_checking: bool = False*)

Bases: *rfslib.abstract_pconnection.PConnection*

Class for SFTP connection. Public interface with an exception of __init__ and close is inherited from PConnection.

__init__(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *username: str*, *password: Optional[str] = None*, *keyfile: str = '~/.ssh/id_rsa'*, *port: int = 22*, *no_host_key_checking: bool = False*)

The constructor of SftpPConnection. Opens SFTP connection, when called. If None password is specified, the key authentication will be used. Otherwise the password authentication will be used.

Parameters

- **settings** – The settings for the super class PConnection.
- **host** – Remote address of the server.
- **port** – Port for the SFTP connection.
- **username** – Remote username
- **password** – Password for a SFTP connection. If None is provided, key authentication will be used.
- **keyfile** – A path to key file.
- **no_host_key_checking** – Specifies, whether remote host key should be verified or not.

_exists(*remote_path*)

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns False.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_isdir(*remote_path*)

Protected method which checks, whether a remote file is a directory.

Parameters **remote_path** – A path of a directory.

Returns True, if remote file is folder. False, if it isn't a folder. Undefined if the file doesn't exist.

_lexists(*remote_path*)
Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns True.

KNOWN BUG: Behavior is undefined in case of broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_listdir(*remote_path*)
Protected method which returns a list of files in the folder including hidden files. Undefined if the remote file doesn't exist or isn't a folder.

Parameters **remote_path** – The remote path of a remote folder.

Returns List of files in the remote folder

_mkdir(*remote_path*)
Protected method which creates a new directory. Behavior is undefined if remote folder already exist, or destination folder doesn't exist.

Parameters **remote_path** – A path of a new remote directory.

_pull(*remote_path*, *local_path*)
Protected method which downloads/pulls a nondirectory file from a remote storage to a local storage in the binary form. Behavior is undefined if source file or destination folder doesn't exist.

Parameters

- **remote_path** – Path of a remote file to download.
- **local_path** – Path of a local file, where to download/pull a remote file or local file already exists.

_push(*local_path*, *remote_path*)
Protected method which uploads/pushes a nondirectory file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder or source file doesn't exist, source is directory or remote file already exists.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

_rename(*old_name*, *new_name*)
Protected method which renames/moves a file. Behavior is undefined, if *new_name* file exists or *old_name* file doesn't exist.

Parameters

- **old_name** – Remote path a file to move.
- **new_name** – Remote path to which move the file.

_rmdir(*remote_path*)
Protected method which removes an empty remote directory. Behavior is undefined if remote directory doesn't exist or it isn't empty.

Parameters **remote_path** – Path of an empty remote directory to delete.

_stat(*remote_path*)
Protected method which returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file. Undefined behavior if remote file doesn't exist or it is a broken symlink.

Parameters **remote_path** – Path of a remote file.

Returns The function returns os.stat_result like object, which is further parsed by _stat_unpack function. For more details please see source code.

_unlink(*remote_path*)
Protected method which removes a nondirectory file. Behavior is undefined if remote file doesn't exist or is a directory.

Parameters **remote_path** – Path of a remote regular file to delete.

close()
Method to close the opened connection.

cp(*old_names*, *new_name*, *recursive=False*)

dcp(*old_names*, *target_dir*, *recursive=False*)

dmv(*old_names*, *target_dir*)

exists(*remote_path: str*) → bool
Method which checks, whether a remote file exist. Returns False for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

fcp(*old_name*, *new_name*)

find(*remote_path*, *child_first=False*)

fmv(*old_name*, *new_name*)

get_default_dmask() → int
Returns default_dmask settings. For moe details see p_connection_settings.

get_default_fmask() → int
Returns default_fmask settings. For more details see p_connection_settings.

get_settings() → *rfslib.abstract_pconnection.p_connection_settings*
The procedure sets all generic settings for PConnection.

Returns A p_connection_settings object with all generic settings of PConnection.

isdir(*remote_path*)

lexists(*remote_path*)
Method which checks, whether a remote file exist. Returns True for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

listdir(*remote_path*)

ls(*remote_path*)

lstat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Doesn't follow symlinks.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)
Uploads/pushes a file from a local storage to a remote storage in the binary form.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

rename(*old_name*, *new_name*)

rm(*remote_path*, *recursive=False*)

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

set_settings(*settings:* rfslib.abstract_pconnection.p_connection_settings)
The procedure sets all generic settings for PConnection.

Parameters **settings** – A p_connection_settings object with all generic settings for PConnection. If some attribute in object is missing, no operation will be done with it.

stat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

CHAPTER

THREE

RFSLIB.FTP_PCONNECTION MODULE

class rfslib.ftp_pconnection.**FtpPConnection**(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *username: str*, *password: str*, *port: int = 21*, *tls: bool = False*, *passive_mode: bool = False*, *debug_level: int = 1*, *connection_encoding: str = 'UTF8'*)

Bases: *rfslib.abstract_pconnection.PConnection*

Class for FTP connection. Public interface with an exception of __init__ and close is inherited from PConnection.

__init__(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *username: str*, *password: str*, *port: int = 21*, *tls: bool = False*, *passive_mode: bool = False*, *debug_level: int = 1*, *connection_encoding: str = 'UTF8'*)

The constructor of FtpPConnection.

Parameters

- **settings** – The settings for the super class PConnection.
- **host** – Remote address of the server.
- **port** – Port for a connection.
- **username** – Remote username.
- **password** – Remote password.
- **tls** – Enables TLS.
- **passive_mode** – Enables passive mode of FTP connection.
- **debug_level** – Specifies how much logs should be generated. 0 - almost non, 1 - more, 2 - log almost everything
- **connection_encoding** – Encoding used for a connection.

_exists(*remote_path*)

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns False.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_isdir(*remote_path*)

Protected method which checks, whether a remote file is a directory.

Parameters **remote_path** – A path of a directory.

Returns True, if remote file is folder. False, if it isn't a folder. Undefined if the file doesn't exist.

_lexists(*remote_path*)
Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns True.

KNOWN BUG: Behavior is undefined in case of broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_listdir(*remote_path*)
Protected method which returns a list of files in the folder including hidden files. Undefined if the remote file doesn't exist or isn't a folder.

Parameters **remote_path** – The remote path of a remote folder.

Returns List of files in the remote folder

_mkdir(*remote_path*)
Protected method which creates a new directory. Behavior is undefined if remote folder already exist, or destination folder doesn't exist.

Parameters **remote_path** – A path of a new remote directory.

_pull(*remote_path*, *local_path*)
Protected method which downloads/pulls a nondirectory file from a remote storage to a local storage in the binary form. Behavior is undefined if source file or destination folder doesn't exist.

Parameters

- **remote_path** – Path of a remote file to download.
- **local_path** – Path of a local file, where to download/pull a remote file or local file already exists.

_push(*local_path*, *remote_path*)
Protected method which uploads/pushes a nondirectory file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder or source file doesn't exist, source is directory or remote file already exists.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

_rename(*old_name*, *new_name*)
Protected method which renames/moves a file. Behavior is undefined, if *new_name* file exists or *old_name* file doesn't exist.

Parameters

- **old_name** – Remote path a file to move.
- **new_name** – Remote path to which move the file.

_rmdir(*remote_path*)
Protected method which removes an empty remote directory. Behavior is undefined if remote directory doesn't exist or it isn't empty.

Parameters **remote_path** – Path of an empty remote directory to delete.

_stat(*remote_path*)
Protected method which returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file. Undefined behavior if remote file doesn't exist or it is a broken symlink.

Parameters **remote_path** – Path of a remote file.

Returns The function returns os.stat_result like object, which is further parsed by _stat_unpack function. For more details please see source code.

_unlink(*remote_path*)
Protected method which removes a nondirectory file. Behavior is undefined if remote file doesn't exist or is a directory.

Parameters **remote_path** – Path of a remote regular file to delete.

close()
Method to close the opened connection.

cp(*old_names*, *new_name*, *recursive=False*)

dcp(*old_names*, *target_dir*, *recursive=False*)

dmv(*old_names*, *target_dir*)

exists(*remote_path: str*) → bool
Method which checks, whether a remote file exist. Returns False for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

fcp(*old_name*, *new_name*)

find(*remote_path*, *child_first=False*)

fmv(*old_name*, *new_name*)

get_default_dmask() → int
Returns default_dmask settings. For moe details see p_connection_settings.

get_default_fmask() → int
Returns default_fmask settings. For more details see p_connection_settings.

get_settings() → *rfslib.abstract_pconnection.p_connection_settings*
The procedure sets all generic settings for PConnection.

Returns A p_connection_settings object with all generic settings of PConnection.

isdir(*remote_path*)

lexists(*remote_path*)
Method which checks, whether a remote file exist. Returns True for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

listdir(*remote_path*)

ls(*remote_path*)

lstat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Doesn't follow symlinks.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)
Uploads/pushes a file from a local storage to a remote storage in the binary form.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

rename(*old_name*, *new_name*)

rm(*remote_path*, *recursive=False*)

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

set_settings(*settings:* rfslib.abstract_pconnection.p_connection_settings)
The procedure sets all generic settings for PConnection.

Parameters **settings** – A p_connection_settings object with all generic settings for PConnection. If some attribute in object is missing, no operation will be done with it.

stat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

CHAPTER

FOUR

RFSLIB.SMB12_PCONNECTION MODULE

class rfslib.smb12_pconnection.**Smb12PConnection**(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *service_name: str*, *username: str*, *password: str*, *port: int = 139*, *use_direct_tcp: bool = False*, *client_name: str = 'RFS'*, *use_ntlm_v1: bool = False*)

Bases: *rfslib.abstract_pconnection.PConnection*

Class for SMB connection version 1 or 2. Public interface with an exception of __init__ and close is inherited from PConnection.

__init__(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *service_name: str*, *username: str*, *password: str*, *port: int = 139*, *use_direct_tcp: bool = False*, *client_name: str = 'RFS'*, *use_ntlm_v1: bool = False*)

The constructor of Smb12PConnection. Opens SMB connection version 1 or 2, when called.

Parameters

- **settings** – The settings for the super class PConnection.
- **host** – Remote address of the server.
- **service_name** – Name of a shared folder.
- **port** – Port for the SMB connection.
- **username** – Remote username.
- **password** – Remote password.
- **use_direct_tcp** – Activates direct tcp mode for SMB.
- **client_name** – Name of this client, which will be sent to a server.
- **use_ntlm_v1** – Enables NTLM version 1 instead of NTLM version 2.

_exists(*remote_path*)

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns False.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_isdir(*remote_path*)

Protected method which checks, whether a remote file is a directory.

Parameters **remote_path** – A path of a directory.

Returns True, if remote file is folder. False, if it isn't a folder. Undefined if the file doesn't exist.

_lexists(*remote_path*)
Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns True.

KNOWN BUG: Behavior is undefined in case of broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_listdir(*remote_path*)
Protected method which returns a list of files in the folder including hidden files. Undefined if the remote file doesn't exist or isn't a folder.

Parameters **remote_path** – The remote path of a remote folder.

Returns List of files in the remote folder

_mkdir(*remote_path*)
Protected method which creates a new directory. Behavior is undefined if remote folder already exist, or destination folder doesn't exist.

Parameters **remote_path** – A path of a new remote directory.

_pull(*remote_path*, *local_path*)
Protected method which downloads/pulls a nondirectory file from a remote storage to a local storage in the binary form. Behavior is undefined if source file or destination folder doesn't exist.

Parameters

- **remote_path** – Path of a remote file to download.
- **local_path** – Path of a local file, where to download/pull a remote file or local file already exists.

_push(*local_path*, *remote_path*)
Protected method which uploads/pushes a nondirectory file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder or source file doesn't exist, source is directory or remote file already exists.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

_rename(*old_name*, *new_name*)
Protected method which renames/moves a file. Behavior is undefined, if *new_name* file exists or *old_name* file doesn't exist.

Parameters

- **old_name** – Remote path a file to move.
- **new_name** – Remote path to which move the file.

_rmdir(*remote_path*)
Protected method which removes an empty remote directory. Behavior is undefined if remote directory doesn't exist or it isn't empty.

Parameters **remote_path** – Path of an empty remote directory to delete.

_stat(*remote_path*)
Protected method which returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file. Undefined behavior if remote file doesn't exist or it is a broken symlink.

Parameters **remote_path** – Path of a remote file.

Returns The function returns os.stat_result like object, which is further parsed by _stat_unpack function. For more details please see source code.

_unlink(*remote_path*)
Protected method which removes a nondirectory file. Behavior is undefined if remote file doesn't exist or is a directory.

Parameters **remote_path** – Path of a remote regular file to delete.

close()
Method to close the opened connection.

cp(*old_names*, *new_name*, *recursive=False*)

dcp(*old_names*, *target_dir*, *recursive=False*)

dmv(*old_names*, *target_dir*)

exists(*remote_path: str*) → bool
Method which checks, whether a remote file exist. Returns False for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

fcp(*old_name*, *new_name*)

find(*remote_path*, *child_first=False*)

fmv(*old_name*, *new_name*)

get_default_dmask() → int
Returns default_dmask settings. For moe details see p_connection_settings.

get_default_fmask() → int
Returns default_fmask settings. For more details see p_connection_settings.

get_settings() → *rfslib.abstract_pconnection.p_connection_settings*
The procedure sets all generic settings for PConnection.

Returns A p_connection_settings object with all generic settings of PConnection.

isdir(*remote_path*)

lexists(*remote_path*)
Method which checks, whether a remote file exist. Returns True for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

listdir(*remote_path*)

ls(*remote_path*)

lstat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Doesn't follow symlinks.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)
Uploads/pushes a file from a local storage to a remote storage in the binary form.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

rename(*old_name*, *new_name*)

rm(*remote_path*, *recursive=False*)

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

set_settings(*settings:* rfslib.abstract_pconnection.p_connection_settings)
The procedure sets all generic settings for PConnection.

Parameters **settings** – A p_connection_settings object with all generic settings for PConnection. If some attribute in object is missing, no operation will be done with it.

stat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

CHAPTER

FIVE

RFSLIB.SMB23_PCONNECTION MODULE

class rfslib.smb23_pconnection.**Smb23PConnection**(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *service_name: str*, *username: str*, *password: str*, *port: int = 445*, *enable_encryption: bool = False*, *dont_require_signing: bool = False*)

Bases: *rfslib.abstract_pconnection.PConnection*

Class for SMB connection version 2 or 3. Public interface with an exception of __init__ and close is inherited from PConnection.

__init__(*settings:* rfslib.abstract_pconnection.p_connection_settings, *host: str*, *service_name: str*, *username: str*, *password: str*, *port: int = 445*, *enable_encryption: bool = False*, *dont_require_signing: bool = False*)

The constructor of Smb23PConnection. Opens SMB connection version 2 or 3, when called.

Parameters

- **settings** – The settings for the super class PConnection.
- **service_name** – Name of a shared folder.
- **host** – Remote address of the server.
- **port** – Port for a SMB connection.
- **username** – Remote username
- **password** – Password for a SMB connection.
- **enable_encryption** – Enables encryption for a SMB3 connection.
- **dont_require_signing** – Disables signing requirement.

_exists(*remote_path*)

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns False.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_isdir(*remote_path*)

Protected method which checks, whether a remote file is a directory.

Parameters **remote_path** – A path of a directory.

Returns True, if remote file is folder. False, if it isn't a folder. Undefined if the file doesn't exist.

_lexists(*remote_path*)
Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns True.

KNOWN BUG: Behavior is undefined in case of broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_listdir(*remote_path*)
Protected method which returns a list of files in the folder including hidden files. Undefined if the remote file doesn't exist or isn't a folder.

Parameters **remote_path** – The remote path of a remote folder.

Returns List of files in the remote folder

_mkdir(*remote_path*)
Protected method which creates a new directory. Behavior is undefined if remote folder already exist, or destination folder doesn't exist.

Parameters **remote_path** – A path of a new remote directory.

_pull(*remote_path*, *local_path*)
Protected method which downloads/pulls a nondirectory file from a remote storage to a local storage in the binary form. Behavior is undefined if source file or destination folder doesn't exist.

Parameters

- **remote_path** – Path of a remote file to download.
- **local_path** – Path of a local file, where to download/pull a remote file or local file already exists.

_push(*local_path*, *remote_path*)
Protected method which uploads/pushes a nondirectory file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder or source file doesn't exist, source is directory or remote file already exists.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

_rename(*old_name*, *new_name*)
Protected method which renames/moves a file. Behavior is undefined, if *new_name* file exists or *old_name* file doesn't exist.

Parameters

- **old_name** – Remote path a file to move.
- **new_name** – Remote path to which move the file.

_rmdir(*remote_path*)
Protected method which removes an empty remote directory. Behavior is undefined if remote directory doesn't exist or it isn't empty.

Parameters **remote_path** – Path of an empty remote directory to delete.

_stat(*remote_path*)
Protected method which returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file. Undefined behavior if remote file doesn't exist or it is a broken symlink.

Parameters **remote_path** – Path of a remote file.

Returns The function returns os.stat_result like object, which is further parsed by _stat_unpack function. For more details please see source code.

_unlink(*remote_path*)
Protected method which removes a nondirectory file. Behavior is undefined if remote file doesn't exist or is a directory.

Parameters **remote_path** – Path of a remote regular file to delete.

close()
Method to close the opened connection.

cp(*old_names*, *new_name*, *recursive=False*)

dcp(*old_names*, *target_dir*, *recursive=False*)

dmv(*old_names*, *target_dir*)

exists(*remote_path: str*) → bool
Method which checks, whether a remote file exist. Returns False for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

fcp(*old_name*, *new_name*)

find(*remote_path*, *child_first=False*)

fmv(*old_name*, *new_name*)

get_default_dmask() → int
Returns default_dmask settings. For moe details see p_connection_settings.

get_default_fmask() → int
Returns default_fmask settings. For more details see p_connection_settings.

get_settings() → *rfslib.abstract_pconnection.p_connection_settings*
The procedure sets all generic settings for PConnection.

Returns A p_connection_settings object with all generic settings of PConnection.

isdir(*remote_path*)

lexists(*remote_path*)
Method which checks, whether a remote file exist. Returns True for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

listdir(*remote_path*)

ls(*remote_path*)

lstat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Doesn't follow symlinks.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)
Uploads/pushes a file from a local storage to a remote storage in the binary form.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

rename(*old_name*, *new_name*)

rm(*remote_path*, *recursive=False*)

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

set_settings(*settings:* rfslib.abstract_pconnection.p_connection_settings)
The procedure sets all generic settings for PConnection.

Parameters **settings** – A p_connection_settings object with all generic settings for PConnection. If some attribute in object is missing, no operation will be done with it.

stat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

rfslib.smb23_pconnection.**config_smb23**(*no_dfs: bool = False*, *disable_secure_negotiate: bool = False*, *dfs_domain_controller: Optional[str] = None*)
The procedure changes global setting for SMB version 2 or 3 across all connection. Don't change value, if any SMB connection version 2 or 3 is active.

Parameters

- **no_dfs** – Disables DFS support - useful as a bug fix.
- **disable_secure_negotiate** – Disables secure negotiate requirement for a SMB connection.
- **dfs_domain_controller** – The DFS domain controller address. Useful in case, when rfstools fails to find it themself.

CHAPTER

SIX

RFSLIB.FS_PCONNECTION MODULE

class rfslib.fs_pconnection.**FsPConnection**(*settings:* rfslib.abstract_pconnection.p_connection_settings)

Bases: *rfslib.abstract_pconnection.PConnection*

Class for operating with local filesystem. Public interface with an exception of __init__ and close is inherited from PConnection.

__init__(*settings:* rfslib.abstract_pconnection.p_connection_settings)

The constructor of FsPConnection.

Parameters **settings** – The settings for super class PConnection.

_exists(*remote_path*)

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns False.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_isdir(*remote_path*)

Protected method which checks, whether a remote file is a directory.

Parameters **remote_path** – A path of a directory.

Returns True, if remote file is folder. False, if it isn't a folder. Undefined if the file doesn't exist.

_lexists(*remote_path*)

Protected method which checks, whether a remote file exist. If the remote file is a broken symlink, it returns True.

KNOWN BUG: Behavior is undefined in case of broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file is exist. False, if remote file doesn't exist

_listdir(*remote_path*)

Protected method which returns a list of files in the folder including hidden files. Undefined if the remote file doesn't exist or isn't a folder.

Parameters **remote_path** – The remote path of a remote folder.

Returns List of files in the remote folder

_mkdir(*remote_path*)

Protected method which creates a new directory. Behavior is undefined if remote folder already exist, or destination folder doesn't exist.

Parameters **remote_path** – A path of a new remote directory.

_pull(*remote_path*, *local_path*)
Protected method which downloads/pulls a nondirectory file from a remote storage to a local storage in the binary form. Behavior is undefined if source file or destination folder doesn't exist.

Parameters

- **remote_path** – Path of a remote file to download.
- **local_path** – Path of a local file, where to download/pull a remote file or local file already exists.

_push(*local_path*, *remote_path*)
Protected method which uploads/pushes a nondirectory file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder or source file doesn't exist, source is directory or remote file already exists.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

_rename(*old_name*, *new_name*)
Protected method which renames/moves a file. Behavior is undefined, if *new_name* file exists or *old_name* file doesn't exist.

Parameters

- **old_name** – Remote path a file to move.
- **new_name** – Remote path to which move the file.

_rmdir(*remote_path*)
Protected method which removes an empty remote directory. Behavior is undefined if remote directory doesn't exist or it isn't empty.

Parameters **remote_path** – Path of an empty remote directory to delete.

_stat(*remote_path*)
Protected method which returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file. Undefined behavior if remote file doesn't exist or it is a broken symlink.

Parameters **remote_path** – Path of a remote file.

Returns The function returns os.stat_result like object, which is further parsed by _stat_unpack function. For more details please see source code.

_unlink(*remote_path*)
Protected method which removes a nondirectory file. Behavior is undefined if remote file doesn't exist or is a directory.

Parameters **remote_path** – Path of a remote regular file to delete.

close()
Method to close the opened connection.

cp(*old_names*, *new_name*, *recursive=False*)

dcp(*old_names*, *target_dir*, *recursive=False*)

dmv(*old_names*, *target_dir*)

exists(*remote_path: str*) → bool
Method which checks, whether a remote file exist. Returns False for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

fcp(*old_name*, *new_name*)

find(*remote_path*, *child_first=False*)

fmv(*old_name*, *new_name*)

get_default_dmask() → int
Returns default_dmask settings. For moe details see p_connection_settings.

get_default_fmask() → int
Returns default_fmask settings. For more details see p_connection_settings.

get_settings() → *rfslib.abstract_pconnection.p_connection_settings*
The procedure sets all generic settings for PConnection.

Returns A p_connection_settings object with all generic settings of PConnection.

isdir(*remote_path*)

lexists(*remote_path*)
Method which checks, whether a remote file exist. Returns True for broken symlinks.

Parameters **remote_path** – Path of a remote file.

Returns True, if remote file exists. False, if remote file doesn't exist.

listdir(*remote_path*)

ls(*remote_path*)

lstat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*
Returns statistics of a file (eg. size, last date modified,...) Doesn't follow symlinks.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)
Uploads/pushes a file from a local storage to a remote storage in the binary form.

Parameters

- **local_path** – Path of a local file to upload.
- **remote_path** – Path on the remote storage, where to upload/push a local file.

rename(*old_name*, *new_name*)

rm(*remote_path*, *recursive=False*)

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

set_settings(*settings:* rfslib.abstract_pconnection.p_connection_settings)
The procedure sets all generic settings for PConnection.

Parameters **settings** – A p_connection_settings object with all generic settings for PConnection. If some attribute in object is missing, no operation will be done with it.

stat(*remote_path: str*) → *rfslib.abstract_pconnection.p_stat_result*

Returns statistics of a file (eg. size, last date modified,...) Follows symlinks to a destination file.

Parameters **remote_path** – Path of a remote file.

Returns An object whose attributes correspond to the attributes of Python's stat structure as returned by os.stat, except that it contains fewer fields.

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

CHAPTER

SEVEN

RFSLIB.PATH_UTILS MODULE

class rfslib.path_utils.**GenericPath**(*path*)

Bases: object

__init__(*path*)

rfslib.path_utils.**add_r_prefix**(*path*)

rfslib.path_utils.**generic_cp**(*conn*, *sources*, *dest*, *recursive=False*)

rfslib.path_utils.**generic_mv**(*conn*, *sources*, *dest*)

rfslib.path_utils.**generic_path_normalize**(*path*)

rfslib.path_utils.**is_remote**(*path*)

rfslib.path_utils.**path_normalize**(*path*)

rfslib.path_utils.**remove_r_prefix**(*path*)

PYTHON MODULE INDEX

r

rfslib.abstract_pconnection, 3
rfslib.fs_pconnection, 25
rfslib.ftp_pconnection, 13
rfslib.path_utils, 29
rfslib.sftp_pconnection, 9
rfslib.smb12_pconnection, 17
rfslib.smb23_pconnection, 21

INDEX

Symbols

__init__() (*rfslib.abstract_pconnection.PConnection method*), 3
__init__() (*rfslib.abstract_pconnection.p_connection_settings method*), 6
__init__() (*rfslib.abstract_pconnection.p_stat_result method*), 6
__init__() (*rfslib.fs_pconnection.FsPConnection method*), 25
__init__() (*rfslib.ftp_pconnection.FtpPConnection method*), 13
__init__() (*rfslib.path_utils.GenericPath method*), 29
__init__() (*rfslib.sftp_pconnection.SftpPConnection method*), 9
__init__() (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
__init__() (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
_exists() (*rfslib.abstract_pconnection.PConnection method*), 3
_exists() (*rfslib.fs_pconnection.FsPConnection method*), 25
_exists() (*rfslib.ftp_pconnection.FtpPConnection method*), 13
_exists() (*rfslib.sftp_pconnection.SftpPConnection method*), 9
_exists() (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
_exists() (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
_isdir() (*rfslib.abstract_pconnection.PConnection method*), 3
_isdir() (*rfslib.fs_pconnection.FsPConnection method*), 25
_isdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 13
_isdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 9
_isdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
_isdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
_lexists() (*rfslib.abstract_pconnection.PConnection method*), 3
_lexists() (*rfslib.fs_pconnection.FsPConnection method*), 25
_lexists() (*rfslib.ftp_pconnection.FtpPConnection method*), 13
_lexists() (*rfslib.sftp_pconnection.SftpPConnection method*), 9
_lexists() (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
_lexists() (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
_listdir() (*rfslib.abstract_pconnection.PConnection method*), 3
_listdir() (*rfslib.fs_pconnection.FsPConnection method*), 25
_listdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 14
_listdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 10
_listdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 18
_listdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 22
_mkdir() (*rfslib.abstract_pconnection.PConnection method*), 3
_mkdir() (*rfslib.fs_pconnection.FsPConnection method*), 25
_mkdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 14
_mkdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 10
_mkdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 18
_mkdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 22
_pull() (*rfslib.abstract_pconnection.PConnection method*), 4
_pull() (*rfslib.fs_pconnection.FsPConnection method*), 25
_pull() (*rfslib.ftp_pconnection.FtpPConnection method*), 14

_pull() (*rfslib.sftp_pconnection.SftpPConnection method*), 10
_pull() (*rfslib.smb12_pconnection.Smb12PConnection method*), 18
_pull() (*rfslib.smb23_pconnection.Smb23PConnection method*), 22
_push() (*rfslib.abstract_pconnection.PConnection method*), 4
_push() (*rfslib.fs_pconnection.FsPConnection method*), 26
_push() (*rfslib.ftp_pconnection.FtpPConnection method*), 14
_push() (*rfslib.sftp_pconnection.SftpPConnection method*), 10
_push() (*rfslib.smb12_pconnection.Smb12PConnection method*), 18
_push() (*rfslib.smb23_pconnection.Smb23PConnection method*), 22
_rename() (*rfslib.abstract_pconnection.PConnection method*), 4
_rename() (*rfslib.fs_pconnection.FsPConnection method*), 26
_rename() (*rfslib.ftp_pconnection.FtpPConnection method*), 14
_rename() (*rfslib.sftp_pconnection.SftpPConnection method*), 10
_rename() (*rfslib.smb12_pconnection.Smb12PConnection method*), 18
_rename() (*rfslib.smb23_pconnection.Smb23PConnection method*), 22
_rmdir() (*rfslib.abstract_pconnection.PConnection method*), 4
_rmdir() (*rfslib.fs_pconnection.FsPConnection method*), 26
_rmdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 14
_rmdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 10
_rmdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 18
_rmdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 22
_stat() (*rfslib.abstract_pconnection.PConnection method*), 4
_stat() (*rfslib.fs_pconnection.FsPConnection method*), 26
_stat() (*rfslib.ftp_pconnection.FtpPConnection method*), 14
_stat() (*rfslib.sftp_pconnection.SftpPConnection method*), 10
_stat() (*rfslib.smb12_pconnection.Smb12PConnection method*), 18
_stat() (*rfslib.smb23_pconnection.Smb23PConnection method*), 22
_unlink() (*rfslib.abstract_pconnection.PConnection method*), 4
_unlink() (*rfslib.fs_pconnection.FsPConnection method*), 26
_unlink() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
_unlink() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
_unlink() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
_unlink() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

A

add_r_prefix() (*in module rfslib.path_utils*), 29

C

close() (*rfslib.abstract_pconnection.PConnection method*), 4
close() (*rfslib.fs_pconnection.FsPConnection method*), 26
close() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
close() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
close() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
close() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
config_smb23() (*in module rfslib.smb23_pconnection*), 24
cp() (*rfslib.abstract_pconnection.PConnection method*), 4
cp() (*rfslib.fs_pconnection.FsPConnection method*), 26
cp() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
cp() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
cp() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
cp() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

D

dcp() (*rfslib.abstract_pconnection.PConnection method*), 4
dcp() (*rfslib.fs_pconnection.FsPConnection method*), 26
dcp() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
dcp() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
dcp() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19

dcp() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
default_dmask (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
default_fmask (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
direct_write (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
dmv() (*rfslib.abstract_pconnection.PConnection method*), 4
dmv() (*rfslib.fs_pconnection.FsPConnection method*), 26
dmv() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
dmv() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
dmv() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
dmv() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

E

exists() (*rfslib.abstract_pconnection.PConnection method*), 4
exists() (*rfslib.fs_pconnection.FsPConnection method*), 26
exists() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
exists() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
exists() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
exists() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

F

fcp() (*rfslib.abstract_pconnection.PConnection method*), 5
fcp() (*rfslib.fs_pconnection.FsPConnection method*), 27
fcp() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
fcp() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
fcp() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
fcp() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
find() (*rfslib.abstract_pconnection.PConnection method*), 5
find() (*rfslib.fs_pconnection.FsPConnection method*), 27
find() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
find() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
find() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
find() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
fmtimings() (*rfslib.abstract_pconnection.PConnection method*), 5
fmtimings() (*rfslib.fs_pconnection.FsPConnection method*), 27
fmv() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
fmv() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
fmv() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
fmv() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
FsPConnection (*class in rfslib.fs_pconnection*), 25
FtpPConnection (*class in rfslib.ftp_pconnection*), 13

G

generic_cp() (*in module rfslib.path_utils*), 29
generic_mv() (*in module rfslib.path_utils*), 29
generic_path_normalize() (*in module rfslib.path_utils*), 29
GenericPath (*class in rfslib.path_utils*), 29
get_default_dmask() (*rfslib.abstract_pconnection.PConnection method*), 5
get_default_dmask() (*rfslib.fs_pconnection.FsPConnection method*), 27
get_default_dmask() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
get_default_dmask() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
get_default_dmask() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
get_default_dmask() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
get_default_fmask() (*rfslib.abstract_pconnection.PConnection method*), 5
get_default_fmask() (*rfslib.fs_pconnection.FsPConnection method*), 27
get_default_fmask() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
get_default_fmask() (*rfslib.sftp_pconnection.SftpPConnection method*), 11

get_default_fmask() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
get_default_fmask() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
get_settings() (*rfslib.abstract_pconnection.PConnection method*), 5
get_settings() (*rfslib.fs_pconnection.FsPConnection method*), 27
get_settings() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
get_settings() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
get_settings() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
get_settings() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

I

is_remote() (*in module rfslib.path_utils*), 29
isdir() (*rfslib.abstract_pconnection.PConnection method*), 5
isdir() (*rfslib.fs_pconnection.FsPConnection method*), 27
isdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
isdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
isdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
isdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

L

lexists() (*rfslib.abstract_pconnection.PConnection method*), 5
lexists() (*rfslib.fs_pconnection.FsPConnection method*), 27
lexists() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
lexists() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
lexists() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
lexists() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
listdir() (*rfslib.abstract_pconnection.PConnection method*), 5
listdir() (*rfslib.fs_pconnection.FsPConnection method*), 27
listdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
listdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
listdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
listdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
local_crlf (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
local_encoding (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
ls() (*rfslib.abstract_pconnection.PConnection method*), 5
ls() (*rfslib.fs_pconnection.FsPConnection method*), 27
ls() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
ls() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
ls() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
ls() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
lstat() (*rfslib.abstract_pconnection.PConnection method*), 5
lstat() (*rfslib.fs_pconnection.FsPConnection method*), 27
lstat() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
lstat() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
lstat() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
lstat() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

M

mkdir() (*rfslib.abstract_pconnection.PConnection method*), 5
mkdir() (*rfslib.fs_pconnection.FsPConnection method*), 27
mkdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
mkdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
mkdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
mkdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
module
    rfslib.abstract_pconnection, 3
    rfslib.fs_pconnection, 25
    rfslib.ftp_pconnection, 13
    rfslib.path_utils, 29
    rfslib.sftp_pconnection, 9
    rfslib.smb12_pconnection, 17

rfslib.smb23_pconnection, 21
mv() (*rfslib.abstract_pconnection.PConnection method*), 5
mv() (*rfslib.fs_pconnection.FsPConnection method*), 27
mv() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
mv() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
mv() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
mv() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23

P

p_connection_settings (*class in rfslib.abstract_pconnection*), 6
p_stat_result (*class in rfslib.abstract_pconnection*), 6
path_normalize() (*in module rfslib.path_utils*), 29
PConnection (*class in rfslib.abstract_pconnection*), 3
pmkdir() (*rfslib.abstract_pconnection.PConnection method*), 5
pmkdir() (*rfslib.fs_pconnection.FsPConnection method*), 27
pmkdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 15
pmkdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 11
pmkdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 19
pmkdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 23
pull() (*rfslib.abstract_pconnection.PConnection method*), 5
pull() (*rfslib.fs_pconnection.FsPConnection method*), 27
pull() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
pull() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
pull() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
pull() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24
push() (*rfslib.abstract_pconnection.PConnection method*), 5
push() (*rfslib.fs_pconnection.FsPConnection method*), 27
push() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
push() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
push() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
push() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24

R

remote_crlf (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
remote_encoding (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
remove_r_prefix() (*in module rfslib.path_utils*), 29
rename() (*rfslib.abstract_pconnection.PConnection method*), 5
rename() (*rfslib.fs_pconnection.FsPConnection method*), 27
rename() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
rename() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
rename() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
rename() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24
rfslib.abstract_pconnection module, 3
rfslib.fs_pconnection module, 25
rfslib.ftp_pconnection module, 13
rfslib.path_utils module, 29
rfslib.sftp_pconnection module, 9
rfslib.smb12_pconnection module, 17
rfslib.smb23_pconnection module, 21
rm() (*rfslib.abstract_pconnection.PConnection method*), 5
rm() (*rfslib.fs_pconnection.FsPConnection method*), 27
rm() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
rm() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
rm() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
rm() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24
rmdir() (*rfslib.abstract_pconnection.PConnection method*), 5
rmdir() (*rfslib.fs_pconnection.FsPConnection method*), 27
rmdir() (*rfslib.ftp_pconnection.FtpPConnection method*), 16

rmdir() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
rmdir() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
rmdir() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24
rpull() (*rfslib.abstract_pconnection.PConnection method*), 5
rpull() (*rfslib.fs_pconnection.FsPConnection method*), 27
rpull() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
rpull() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
rpull() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
rpull() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24
rpush() (*rfslib.abstract_pconnection.PConnection method*), 5
rpush() (*rfslib.fs_pconnection.FsPConnection method*), 27
rpush() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
rpush() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
rpush() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
rpush() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24

S

set_settings() (*rfslib.abstract_pconnection.PConnection method*), 5
set_settings() (*rfslib.fs_pconnection.FsPConnection method*), 27
set_settings() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
set_settings() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
set_settings() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
set_settings() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24
SftpPConnection (*class in rfslib.sftp_pconnection*), 9
skip_validation (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
Smb12PConnection (*class in rfslib.smb12_pconnection*), 17
Smb23PConnection (*class in rfslib.smb23_pconnection*), 21
st_atime (*rfslib.abstract_pconnection.p_stat_result attribute*), 7
st_gid (*rfslib.abstract_pconnection.p_stat_result attribute*), 7
st_mode (*rfslib.abstract_pconnection.p_stat_result attribute*), 7
st_mtime (*rfslib.abstract_pconnection.p_stat_result attribute*), 7
st_nlink (*rfslib.abstract_pconnection.p_stat_result attribute*), 7
st_size (*rfslib.abstract_pconnection.p_stat_result attribute*), 7
st_uid (*rfslib.abstract_pconnection.p_stat_result attribute*), 7
stat() (*rfslib.abstract_pconnection.PConnection method*), 6
stat() (*rfslib.fs_pconnection.FsPConnection method*), 28
stat() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
stat() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
stat() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
stat() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24

T

text_transmission (*rfslib.abstract_pconnection.p_connection_settings attribute*), 6
touch() (*rfslib.abstract_pconnection.PConnection method*), 6
touch() (*rfslib.fs_pconnection.FsPConnection method*), 28
touch() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
touch() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
touch() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
touch() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24

U

unlink() (*rfslib.abstract_pconnection.PConnection method*), 6
unlink() (*rfslib.fs_pconnection.FsPConnection method*), 28
unlink() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
unlink() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
unlink() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20

unlink() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24

X

xls() (*rfslib.abstract_pconnection.PConnection method*), 6
xls() (*rfslib.fs_pconnection.FsPConnection method*), 28
xls() (*rfslib.ftp_pconnection.FtpPConnection method*), 16
xls() (*rfslib.sftp_pconnection.SftpPConnection method*), 12
xls() (*rfslib.smb12_pconnection.Smb12PConnection method*), 20
xls() (*rfslib.smb23_pconnection.Smb23PConnection method*), 24