rfstools

Release 1.8.1

Přemysl Šťastný

Apr 08, 2024

CONTENTS:

This is developer documentation of rfstools. For user documentation, use please README.md in base repository.

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 MODULE

class rfslib.**pconnection_settings**

Bases: object

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

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.

CHAPTER

TWO

RFSLIB.ABSTRACT_PCONNECTION MODULE

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

Bases: ABC

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. The function is DEPRECATED and will be substituted using stat or lstat.

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. It might contain '.' and '..'. Undefined if the remote file doesn't exist or isn't a folder.

Parameters
remote_path – The remote path of a remote folder.

Returns
A 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*) → 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: List[str]*, *new_name: str*, *recursive: bool = False*)

dcp(*old_names: List[str]*, *target_dir: str*, *recursive: bool = False*)

dmv(*old_names: List[str]*, *target_dir: str*)

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: str*, *new_name: str*)

find(*remote_path: str*, *child_first: bool = False*) → List[str]
A public method which returns DFS (depth-first search) of remote_path including hidden files. It never returns '.' or '..'.

Parameters
child_first – If True, childs of a directory will be returned before the directory itself.

Returns
The result of DFS as a list of remote_paths.

fmv(*old_name: str*, *new_name: str*)

get_default_dmask() → int
Returns default_dmask settings. For more details see pconnection_settings.

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

get_settings() → *pconnection_settings*
The procedure sets all generic settings for PConnection.

Returns
A pconnection_settings object with all generic settings of PConnection.

isdir(*remote_path: str*)
A public method, which checks, whether there is an folder on remote_path. If yes, true is returned. Otherwise false.

Parameters
remote_path – A path, where to check, whether there is an folder.

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: str*)

Public method which returns a list of files in the folder including hidden files. It never returns '.' or '..'.

Parameters
remote_path – The remote path of a remote folder.

Returns
A list of files in the remote folder.

ls(*remote_path: str*)

lstat(*remote_path: str*) → *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: str*)

A public method, which creates a folder. If directory can't be created, because a file already exist, an exception is raised. No other directories on path will be created and if any of them is missing, an exception is raised.

Parameters
remote_path – A path, where to create a new directory.

mv(*old_names: List[str]*, *new_name: str*)

pmkdir(*remote_path: str*)

pull(*remote_path: str*, *local_path: str*)

push(*local_path: str*, *remote_path: str*)

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: str*, *new_name: str*)

rm(*remote_path: str*, *recursive: bool = False*)

rmdir(*remote_path: str*)

rpull(*remote_path: str*, *local_path: str*)

rpush(*local_path: str*, *remote_path: str*)

set_settings(*settings:* pconnection_settings)

The procedure sets all generic settings for PConnection.

Parameters
settings – A pconnection_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*) → *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: str*)

unlink(*remote_path: str*)

xls(*remote_path: str*)

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.

st_atime: int = None

This is the time of the last access of file data.

st_gid: int = None

This field contains the ID of the group owner of the file.

st_mode: int = None

This field contains the file type and mode.

st_mtime: int = None

This is the time of last modification of file data.

st_nlink: int = None

This field contains the number of hard links to the file.

st_size: int = 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: int = None

This field contains the user ID of the owner of the file.

CHAPTER

THREE

RFSLIB.SFTP_PCONNECTION MODULE

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

Bases: *PConnection*

Class for SFTP connection. Public interface with an exception of __init__ and close is inherited from 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. The function is DEPRECATED and will be substituted using stat or lstat.

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. It might contain '.' and '..'. Undefined if the remote file doesn't exist or isn't a folder.

Parameters
remote_path – The remote path of a remote folder.

Returns
A 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: List[str]*, *new_name: str*, *recursive: bool = False*)

dcp(*old_names: List[str]*, *target_dir: str*, *recursive: bool = False*)

dmv(*old_names: List[str]*, *target_dir: str*)

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: str*, *new_name: str*)

find(*remote_path: str*, *child_first: bool = False*) → List[str]

A public method which returns DFS (depth-first search) of remote_path including hidden files. It never returns '.' or '..'.

Parameters
child_first – If True, childs of a directory will be returned before the directory itself.

Returns
The result of DFS as a list of remote_paths.

fmv(*old_name: str*, *new_name: str*)

get_default_dmask() → int

Returns default_dmask settings. For more details see pconnection_settings.

get_default_fmask() → int

Returns default_fmask settings. For more details see pconnection_settings.

get_settings() → *pconnection_settings*

The procedure sets all generic settings for PConnection.

Returns
A pconnection_settings object with all generic settings of PConnection.

isdir(*remote_path: str*)

A public method, which checks, whether there is an folder on remote_path. If yes, true is returned. Otherwise false.

Parameters
remote_path – A path, where to check, whether there is an folder.

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: str*)

Public method which returns a list of files in the folder including hidden files. It never returns '.' or '..'.

Parameters
remote_path – The remote path of a remote folder.

Returns
A list of files in the remote folder.

ls(*remote_path: str*)

lstat(*remote_path: str*) → *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: str*)

A public method, which creates a folder. If directory can't be created, because a file already exist, an exception is raised. No other directories on path will be created and if any of them is missing, an exception is raised.

Parameters
remote_path – A path, where to create a new directory.

mv(*old_names: List[str]*, *new_name: str*)

pmkdir(*remote_path: str*)

pull(*remote_path: str*, *local_path: str*)

push(*local_path: str*, *remote_path: str*)

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: str*, *new_name: str*)

rm(*remote_path: str*, *recursive: bool = False*)

rmdir(*remote_path: str*)

rpull(*remote_path: str*, *local_path: str*)

rpush(*local_path: str*, *remote_path: str*)

set_settings(*settings:* pconnection_settings)

The procedure sets all generic settings for PConnection.

Parameters

settings – A pconnection_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*) → *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: str*)

unlink(*remote_path: str*)

xls(*remote_path: str*)

CHAPTER

FOUR

RFSLIB.FTP_PCONNECTION MODULE

class rfslib.ftp_pconnection.**FtpPConnection**(*settings:* pconnection_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'*, *dont_use_list_a: bool = False*)

Bases: *PConnection*

Class for FTP connection. Public interface with an exception of __init__ and close is inherited from 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. The function is DEPRECATED and will be substituted using stat or lstat.

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. It might contain '.' and '..'. Undefined if the remote file doesn't exist or isn't a folder.

Parameters
remote_path – The remote path of a remote folder.

Returns
A 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: List[str]*, *new_name: str*, *recursive: bool = False*)

dcp(*old_names: List[str]*, *target_dir: str*, *recursive: bool = False*)

dmv(*old_names: List[str]*, *target_dir: str*)

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: str*, *new_name: str*)

find(*remote_path: str*, *child_first: bool = False*) → List[str]

A public method which returns DFS (depth-first search) of remote_path including hidden files. It never returns '.' or '..'.

Parameters
child_first – If True, childs of a directory will be returned before the directory itself.

Returns
The result of DFS as a list of remote_paths.

fmv(*old_name: str*, *new_name: str*)

get_default_dmask() → int

Returns default_dmask settings. For more details see pconnection_settings.

get_default_fmask() → int

Returns default_fmask settings. For more details see pconnection_settings.

get_settings() → *pconnection_settings*

The procedure sets all generic settings for PConnection.

Returns
A pconnection_settings object with all generic settings of PConnection.

isdir(*remote_path: str*)

A public method, which checks, whether there is an folder on remote_path. If yes, true is returned. Otherwise false.

Parameters
remote_path – A path, where to check, whether there is an folder.

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: str*)

Public method which returns a list of files in the folder including hidden files. It never returns '.' or '..'.

Parameters
remote_path – The remote path of a remote folder.

Returns
A list of files in the remote folder.

ls(*remote_path: str*)

lstat(*remote_path: str*) → *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: str*)

A public method, which creates a folder. If directory can't be created, because a file already exist, an exception is raised. No other directories on path will be created and if any of them is missing, an exception is raised.

Parameters
remote_path – A path, where to create a new directory.

mv(*old_names: List[str]*, *new_name: str*)

pmkdir(*remote_path: str*)

pull(*remote_path: str*, *local_path: str*)

push(*local_path: str*, *remote_path: str*)

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: str*, *new_name: str*)

rm(*remote_path: str*, *recursive: bool = False*)

rmdir(*remote_path: str*)

rpull(*remote_path: str*, *local_path: str*)

rpush(*local_path: str*, *remote_path: str*)

set_settings(*settings:* pconnection_settings)

The procedure sets all generic settings for PConnection.

Parameters

settings – A pconnection_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*) → *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: str*)

unlink(*remote_path: str*)

xls(*remote_path: str*)

CHAPTER

FIVE

RFSLIB.SMB12_PCONNECTION MODULE

class rfslib.smb12_pconnection.**Smb12PConnection**(*settings:* pconnection_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: *PConnection*

Class for SMB connection version 1 or 2. Public interface with an exception of __init__ and close is inherited from 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. The function is DEPRECATED and will be substituted using stat or lstat.

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. It might contain '.' and '..'. Undefined if the remote file doesn't exist or isn't a folder.

Parameters
remote_path – The remote path of a remote folder.

Returns
A 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: List[str]*, *new_name: str*, *recursive: bool = False*)

dcp(*old_names: List[str]*, *target_dir: str*, *recursive: bool = False*)

dmv(*old_names: List[str]*, *target_dir: str*)

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: str*, *new_name: str*)

find(*remote_path: str*, *child_first: bool = False*) → List[str]

A public method which returns DFS (depth-first search) of remote_path including hidden files. It never returns '.' or '..'.

Parameters
child_first – If True, childs of a directory will be returned before the directory itself.

Returns
The result of DFS as a list of remote_paths.

fmv(*old_name: str*, *new_name: str*)

get_default_dmask() → int

Returns default_dmask settings. For more details see pconnection_settings.

get_default_fmask() → int

Returns default_fmask settings. For more details see pconnection_settings.

get_settings() → *pconnection_settings*

The procedure sets all generic settings for PConnection.

Returns
A pconnection_settings object with all generic settings of PConnection.

isdir(*remote_path: str*)

A public method, which checks, whether there is an folder on remote_path. If yes, true is returned. Otherwise false.

Parameters
remote_path – A path, where to check, whether there is an folder.

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: str*)

Public method which returns a list of files in the folder including hidden files. It never returns '.' or '..'.

Parameters
remote_path – The remote path of a remote folder.

Returns
A list of files in the remote folder.

ls(*remote_path: str*)

lstat(*remote_path: str*) → *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: str*)

A public method, which creates a folder. If directory can't be created, because a file already exist, an exception is raised. No other directories on path will be created and if any of them is missing, an exception is raised.

Parameters
remote_path – A path, where to create a new directory.

mv(*old_names: List[str]*, *new_name: str*)

pmkdir(*remote_path: str*)

pull(*remote_path: str*, *local_path: str*)

push(*local_path: str*, *remote_path: str*)

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: str*, *new_name: str*)

rm(*remote_path: str*, *recursive: bool = False*)

rmdir(*remote_path: str*)

rpull(*remote_path: str*, *local_path: str*)

rpush(*local_path: str*, *remote_path: str*)

set_settings(*settings:* pconnection_settings)

The procedure sets all generic settings for PConnection.

Parameters

settings – A pconnection_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*) → *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: str*)

unlink(*remote_path: str*)

xls(*remote_path: str*)

CHAPTER

SIX

RFSLIB.SMB23_PCONNECTION MODULE

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

Bases: *PConnection*

Class for SMB connection version 2 or 3. Public interface with an exception of __init__ and close is inherited from 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. The function is DEPRECATED and will be substituted using stat or lstat.

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. It might contain '.' and '..'. Undefined if the remote file doesn't exist or isn't a folder.

Parameters
remote_path – The remote path of a remote folder.

Returns
A 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: List[str]*, *new_name: str*, *recursive: bool = False*)

dcp(*old_names: List[str]*, *target_dir: str*, *recursive: bool = False*)

dmv(*old_names: List[str]*, *target_dir: str*)

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: str*, *new_name: str*)

find(*remote_path: str*, *child_first: bool = False*) → List[str]

A public method which returns DFS (depth-first search) of remote_path including hidden files. It never returns '.' or '..'.

Parameters
child_first – If True, childs of a directory will be returned before the directory itself.

Returns
The result of DFS as a list of remote_paths.

fmv(*old_name: str*, *new_name: str*)

get_default_dmask() → int

Returns default_dmask settings. For more details see pconnection_settings.

get_default_fmask() → int

Returns default_fmask settings. For more details see pconnection_settings.

get_settings() → *pconnection_settings*

The procedure sets all generic settings for PConnection.

Returns
A pconnection_settings object with all generic settings of PConnection.

isdir(*remote_path: str*)

A public method, which checks, whether there is an folder on remote_path. If yes, true is returned. Otherwise false.

Parameters
remote_path – A path, where to check, whether there is an folder.

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: str*)

Public method which returns a list of files in the folder including hidden files. It never returns '.' or '..'.

Parameters
remote_path – The remote path of a remote folder.

Returns
A list of files in the remote folder.

ls(*remote_path: str*)

lstat(*remote_path: str*) → *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: str*)

A public method, which creates a folder. If directory can't be created, because a file already exist, an exception is raised. No other directories on path will be created and if any of them is missing, an exception is raised.

Parameters
remote_path – A path, where to create a new directory.

mv(*old_names: List[str]*, *new_name: str*)

pmkdir(*remote_path: str*)

pull(*remote_path: str*, *local_path: str*)

push(*local_path: str*, *remote_path: str*)

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: str*, *new_name: str*)

rm(*remote_path: str*, *recursive: bool = False*)

rmdir(*remote_path: str*)

rpull(*remote_path: str*, *local_path: str*)

rpush(*local_path: str*, *remote_path: str*)

set_settings(*settings:* pconnection_settings)

The procedure sets all generic settings for PConnection.

Parameters

settings – A pconnection_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*) → *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: str*)

unlink(*remote_path: str*)

xls(*remote_path: str*)

rfslib.smb23_pconnection.**config_smb23**(*username: str = None*, *password: str = None*, *no_dfs: bool = False*, *disable_secure_negotiate: bool = False*, *dfs_domain_controller: str = None*, *auth_protocol: str = 'negotiate'*)

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

- **username** – Optional default username used when creating a new SMB session.
- **password** – Optional default password used when creating a new SMB session.
- **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.
- **auth_protocol** – The protocol to use for authentication. Possible values are 'negotiate', 'ntlm' or 'kerberos'. Defaults to 'negotiate'.

CHAPTER

SEVEN

RFSLIB.FS_PCONNECTION MODULE

class rfslib.fs_pconnection.**FsPConnection**(*settings:* pconnection_settings)

Bases: *PConnection*

Class for operating with local filesystem. Public interface with an exception of __init__ and close is inherited from 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. The function is DEPRECATED and will be substituted using stat or lstat.

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. It might contain '.' and '..'. Undefined if the remote file doesn't exist or isn't a folder.

Parameters
remote_path – The remote path of a remote folder.

Returns
A 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: List[str]*, *new_name: str*, *recursive: bool = False*)

dcp(*old_names: List[str]*, *target_dir: str*, *recursive: bool = False*)

dmv(*old_names: List[str]*, *target_dir: str*)

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: str*, *new_name: str*)

find(*remote_path: str*, *child_first: bool = False*) → List[str]
A public method which returns DFS (depth-first search) of remote_path including hidden files. It never returns '.' or '..'.

Parameters
child_first – If True, childs of a directory will be returned before the directory itself.

Returns
The result of DFS as a list of remote_paths.

fmv(*old_name: str*, *new_name: str*)

get_default_dmask() → int
Returns default_dmask settings. For more details see pconnection_settings.

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

get_settings() → *pconnection_settings*
The procedure sets all generic settings for PConnection.

Returns
A pconnection_settings object with all generic settings of PConnection.

isdir(*remote_path: str*)
A public method, which checks, whether there is an folder on remote_path. If yes, true is returned. Otherwise false.

Parameters
remote_path – A path, where to check, whether there is an folder.

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: str*)

Public method which returns a list of files in the folder including hidden files. It never returns '.' or '..'.

Parameters
remote_path – The remote path of a remote folder.

Returns
A list of files in the remote folder.

ls(*remote_path: str*)

lstat(*remote_path: str*) → *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: str*)

A public method, which creates a folder. If directory can't be created, because a file already exist, an exception is raised. No other directories on path will be created and if any of them is missing, an exception is raised.

Parameters
remote_path – A path, where to create a new directory.

mv(*old_names: List[str]*, *new_name: str*)

pmkdir(*remote_path: str*)

pull(*remote_path: str*, *local_path: str*)

push(*local_path: str*, *remote_path: str*)

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: str*, *new_name: str*)

rm(*remote_path: str*, *recursive: bool = False*)

rmdir(*remote_path: str*)

rpull(*remote_path: str*, *local_path: str*)

rpush(*local_path: str*, *remote_path: str*)

set_settings(*settings:* pconnection_settings)

The procedure sets all generic settings for PConnection.

Parameters
settings – A pconnection_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*) → *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: str*)

unlink(*remote_path: str*)

xls(*remote_path: str*)

CHAPTER

EIGHT

RFSLIB.PATH_UTILS MODULE

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

Bases: object

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*)

CHAPTER

NINE

_RFSTOOLS.ARG_PARSER MODULE

_rfstools.arg_parser.**default_arg_parser**(*description: str = ''*, *wildcard_skipper=False*) → ArgumentParser

A function, which is called in the beginning of all scripts in *bin*. It generates a base parser sceleton, which is later altered by script itself.

Parameters

- **description** – The script description. Basically what the script does.
- **wildcard_skipper** – Adds –ignore-failed-wildcards option.

Returns

A base parser sceleton.

_rfstools.arg_parser.**many_to_one_arg_parser**(*description: str = ''*) → ArgumentParser

_rfstools.arg_parser.**one_arg_parser**(*description: str = ''*) → ArgumentParser

_rfstools.arg_parser.**oneplus_arg_parser**(*description: str = ''*, *wildcard_skipper=False*) → ArgumentParser

CHAPTER

TEN

_RFSTOOLS.ARG_PROCESSOR MODULE

_rfstools.arg_processor.**init**(*arg_parser*, *name*, *vars_to_pass*)

PYTHON MODULE INDEX

INDEX

Symbols

A

C

D

E

F

G

I

L

M

S

T

U

X