
rfslib

Release 1.0.22

Přemysl Šťastný

Aug 03, 2021

CONTENTS:

| | | |
|----------|---|-----------|
| 1 | rfslib.abstract_pconnection module | 3 |
| 2 | rfslib.sftp_pconnection module | 7 |
| 3 | rfslib.ftp_pconnection module | 11 |
| 4 | rfslib.smb12_pconnection module | 15 |
| 5 | rfslib.smb23_pconnection module | 19 |
| 6 | rfslib.fs_pconnection module | 23 |
| 7 | rfslib.path_utils module | 27 |
| | Python Module Index | 29 |
| | Index | 31 |

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.

RFSLIB.ABSTRACT_PCONNECTION MODULE

```
class rfslib.abstract_pconnection.PConnection(**args)
```

Bases: abc.ABC

```
__init__(**args)
    TODO
```

```
abstract _exists(remote_path: str) → bool
```

Protected method which checks, whether a remote file exist.

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 _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.

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 file from a remote storage to a local storage in the binary form. Behavior is undefined if 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 file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder doesn't exist 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 _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.

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

fcp(*old_name, new_name*)**find**(*remote_path, child_first=False*)**fmv**(*old_name, new_name*)**isdir**(*remote_path*)

lexists(*remote_path*)

Method which checks, whether a remote file exist.

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*)

ls(*remote_path*)

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)

rename(*old_name*, *new_name*)

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

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

RFSLIB.SFTP_PCONNECTION MODULE

```
class rfslib.sftp_pconnection.SftpPConnection(**arg)
```

Bases: *rfslib.abstract_pconnection.PConnection*

```
__init__(**arg)  
    TODO
```

```
_exists(remote_path)
```

Protected method which checks, whether a remote file exist.

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

```
_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.

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 file from a remote storage to a local storage in the binary form. Behavior is undefined if 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 file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder doesn't exist 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.

_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.

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

fcp(*old_name*, *new_name*)**find**(*remote_path*, *child_first=False*)**fmv**(*old_name*, *new_name*)**isdir**(*remote_path*)

lexists(*remote_path*)

Method which checks, whether a remote file exist.

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*)

ls(*remote_path*)

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)

rename(*old_name*, *new_name*)

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

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

RFSLIB.FTP_PCONNECTION MODULE

```
class rfslib.ftp_pconnection.FtpPConnection(**arg)
    Bases: rfslib.abstract_pconnection.PConnection
```

```
    __init__(**arg)
        TODO
```

```
    _exists(remote_path)
```

Protected method which checks, whether a remote file exist.

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

```
    _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.

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 file from a remote storage to a local storage in the binary form. Behavior is undefined if 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 file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder doesn't exist 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.

_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.

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

fcp(*old_name*, *new_name*)**find**(*remote_path*, *child_first=False*)**fmv**(*old_name*, *new_name*)**isdir**(*remote_path*)

lexists(*remote_path*)

Method which checks, whether a remote file exist.

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*)

ls(*remote_path*)

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)

rename(*old_name*, *new_name*)

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

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

RFSLIB.SMB12_PCONNECTION MODULE

```
class rfslib.smb12_pconnection.Smb12PConnection(**arg)
```

Bases: *rfslib.abstract_pconnection.PConnection*

```
__init__(**arg)  
    TODO
```

```
_exists(remote_path)
```

Protected method which checks, whether a remote file exist.

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

```
_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.

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 file from a remote storage to a local storage in the binary form. Behavior is undefined if 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 file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder doesn't exist 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.

_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.

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

fcp(*old_name*, *new_name*)**find**(*remote_path*, *child_first=False*)**fmv**(*old_name*, *new_name*)**isdir**(*remote_path*)

lexists(*remote_path*)

Method which checks, whether a remote file exist.

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*)

ls(*remote_path*)

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)

rename(*old_name*, *new_name*)

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

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

RFSLIB.SMB23_PCONNECTION MODULE

```
class rfslib.smb23_pconnection.Smb23PConnection(**arg)
```

Bases: *rfslib.abstract_pconnection.PConnection*

```
__init__(**arg)  
    TODO
```

```
_exists(remote_path)
```

Protected method which checks, whether a remote file exist.

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

```
_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.

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 file from a remote storage to a local storage in the binary form. Behavior is undefined if 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 file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder doesn't exist 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.

_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.

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

fcp(*old_name*, *new_name*)**find**(*remote_path*, *child_first=False*)**fmv**(*old_name*, *new_name*)**isdir**(*remote_path*)

lexists(*remote_path*)

Method which checks, whether a remote file exist.

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*)

ls(*remote_path*)

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)

rename(*old_name*, *new_name*)

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

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

`rfslib.smb23_pconnection.config_smb23(**arg)`

RFSLIB.FS_PCONNECTION MODULE

```
class rfslib.fs_pconnection.FsPConnection(**arg)
    Bases: rfslib.abstract_pconnection.PConnection
```

```
    __init__(**arg)
        TODO
```

```
    _exists(remote_path)
```

Protected method which checks, whether a remote file exist.

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

```
    _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.

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 file from a remote storage to a local storage in the binary form. Behavior is undefined if 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 file from a local storage to a remote storage in the binary form. Behavior is undefined if destination folder doesn't exist 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.

_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.

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

fcp(*old_name*, *new_name*)**find**(*remote_path*, *child_first=False*)**fmv**(*old_name*, *new_name*)**isdir**(*remote_path*)

lexists(*remote_path*)

Method which checks, whether a remote file exist.

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*)

ls(*remote_path*)

mkdir(*remote_path*)

mv(*old_names*, *new_name*)

pmkdir(*remote_path*)

pull(*remote_path*, *local_path*)

push(*local_path*, *remote_path*)

rename(*old_name*, *new_name*)

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

rmdir(*remote_path*)

rpull(*remote_path*, *local_path*)

rpush(*local_path*, *remote_path*)

touch(*remote_path*)

unlink(*remote_path*)

xls(*remote_path*)

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`, [23](#)
- `rfslib.ftp_pconnection`, [11](#)
- `rfslib.path_utils`, [27](#)
- `rfslib.sftp_pconnection`, [7](#)
- `rfslib.smb12_pconnection`, [15](#)
- `rfslib.smb23_pconnection`, [19](#)

Symbols

| | |
|--|--|
| <code>__init__()</code> (<i>rfslib.abstract_pconnection.PConnection</i> method), 3 | <code>_lexists()</code> (<i>rfslib.ftp_pconnection.FtpPConnection</i> method), 11 |
| <code>__init__()</code> (<i>rfslib.fs_pconnection.FsPConnection</i> method), 23 | <code>_lexists()</code> (<i>rfslib.sftp_pconnection.SftpPConnection</i> method), 7 |
| <code>__init__()</code> (<i>rfslib.ftp_pconnection.FtpPConnection</i> method), 11 | <code>_lexists()</code> (<i>rfslib.smb12_pconnection.Smb12PConnection</i> method), 15 |
| <code>__init__()</code> (<i>rfslib.path_utils.GenericPath</i> method), 27 | <code>_lexists()</code> (<i>rfslib.smb23_pconnection.Smb23PConnection</i> method), 19 |
| <code>__init__()</code> (<i>rfslib.sftp_pconnection.SftpPConnection</i> method), 7 | <code>_listdir()</code> (<i>rfslib.abstract_pconnection.PConnection</i> method), 3 |
| <code>__init__()</code> (<i>rfslib.smb12_pconnection.Smb12PConnection</i> method), 15 | <code>_listdir()</code> (<i>rfslib.fs_pconnection.FsPConnection</i> method), 23 |
| <code>__init__()</code> (<i>rfslib.smb23_pconnection.Smb23PConnection</i> method), 19 | <code>_listdir()</code> (<i>rfslib.ftp_pconnection.FtpPConnection</i> method), 11 |
| <code>_exists()</code> (<i>rfslib.abstract_pconnection.PConnection</i> method), 3 | <code>_listdir()</code> (<i>rfslib.sftp_pconnection.SftpPConnection</i> method), 7 |
| <code>_exists()</code> (<i>rfslib.fs_pconnection.FsPConnection</i> method), 23 | <code>_listdir()</code> (<i>rfslib.smb12_pconnection.Smb12PConnection</i> method), 15 |
| <code>_exists()</code> (<i>rfslib.ftp_pconnection.FtpPConnection</i> method), 11 | <code>_listdir()</code> (<i>rfslib.smb23_pconnection.Smb23PConnection</i> method), 19 |
| <code>_exists()</code> (<i>rfslib.sftp_pconnection.SftpPConnection</i> method), 7 | <code>_mkdir()</code> (<i>rfslib.abstract_pconnection.PConnection</i> method), 3 |
| <code>_exists()</code> (<i>rfslib.smb12_pconnection.Smb12PConnection</i> method), 15 | <code>_mkdir()</code> (<i>rfslib.fs_pconnection.FsPConnection</i> method), 23 |
| <code>_exists()</code> (<i>rfslib.smb23_pconnection.Smb23PConnection</i> method), 19 | <code>_mkdir()</code> (<i>rfslib.ftp_pconnection.FtpPConnection</i> method), 11 |
| <code>_isdir()</code> (<i>rfslib.abstract_pconnection.PConnection</i> method), 3 | <code>_mkdir()</code> (<i>rfslib.sftp_pconnection.SftpPConnection</i> method), 7 |
| <code>_isdir()</code> (<i>rfslib.fs_pconnection.FsPConnection</i> method), 23 | <code>_mkdir()</code> (<i>rfslib.smb12_pconnection.Smb12PConnection</i> method), 15 |
| <code>_isdir()</code> (<i>rfslib.ftp_pconnection.FtpPConnection</i> method), 11 | <code>_mkdir()</code> (<i>rfslib.smb23_pconnection.Smb23PConnection</i> method), 19 |
| <code>_isdir()</code> (<i>rfslib.sftp_pconnection.SftpPConnection</i> method), 7 | <code>_pull()</code> (<i>rfslib.abstract_pconnection.PConnection</i> method), 3 |
| <code>_isdir()</code> (<i>rfslib.smb12_pconnection.Smb12PConnection</i> method), 15 | <code>_pull()</code> (<i>rfslib.fs_pconnection.FsPConnection</i> method), 23 |
| <code>_isdir()</code> (<i>rfslib.smb23_pconnection.Smb23PConnection</i> method), 19 | <code>_pull()</code> (<i>rfslib.ftp_pconnection.FtpPConnection</i> method), 11 |
| <code>_lexists()</code> (<i>rfslib.abstract_pconnection.PConnection</i> method), 3 | <code>_pull()</code> (<i>rfslib.sftp_pconnection.SftpPConnection</i> method), 7 |
| <code>_lexists()</code> (<i>rfslib.fs_pconnection.FsPConnection</i> method), 23 | <code>_pull()</code> (<i>rfslib.smb12_pconnection.Smb12PConnection</i> method), 15 |

[_pull\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 19
[_push\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[_push\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[_push\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[_push\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[_push\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[_push\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[_rename\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[_rename\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[_rename\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[_rename\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[_rename\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[_rename\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[_rmdir\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[_rmdir\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[_rmdir\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[_rmdir\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[_rmdir\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[_rmdir\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[_unlink\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[_unlink\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[_unlink\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[_unlink\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[_unlink\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[_unlink\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20

A

[add_r_prefix\(\)](#) (in module [rfslib.path_utils](#)), 27

C

[close\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[close\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[close\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[close\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[close\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[close\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[config_smb23\(\)](#) (in module [rfslib.smb23_pconnection](#)), 21
[cp\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[cp\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[cp\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[cp\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[cp\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[cp\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20

D

[dcp\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[dcp\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[dcp\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[dcp\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[dcp\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[dcp\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[dmv\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[dmv\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[dmv\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[dmv\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[dmv\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[dmv\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20

E

[exists\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4

[exists\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[exists\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[exists\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[exists\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[exists\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20

F

[fcp\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[fcp\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[fcp\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[fcp\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[fcp\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[fcp\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[find\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[find\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[find\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[find\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[find\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[find\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[fmv\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[fmv\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[fmv\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[fmv\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[fmv\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[fmv\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[FsPConnection](#) (class in [rfslib.fs_pconnection](#)), 23
[FtpPConnection](#) (class in [rfslib.ftp_pconnection](#)), 11

G

[generic_cp\(\)](#) (in module [rfslib.path_utils](#)), 27
[generic_mv\(\)](#) (in module [rfslib.path_utils](#)), 27
[generic_path_normalize\(\)](#) (in module [rfslib.path_utils](#)), 27

[GenericPath](#) (class in [rfslib.path_utils](#)), 27

I

[is_remote\(\)](#) (in module [rfslib.path_utils](#)), 27
[isdir\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[isdir\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[isdir\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[isdir\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[isdir\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[isdir\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20

L

[lexists\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 4
[lexists\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 24
[lexists\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 12
[lexists\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 8
[lexists\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 16
[lexists\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 20
[listdir\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 5
[listdir\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 25
[listdir\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 13
[listdir\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 9
[listdir\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 17
[listdir\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 21
[ls\(\)](#) ([rfslib.abstract_pconnection.PConnection](#) method), 5
[ls\(\)](#) ([rfslib.fs_pconnection.FsPConnection](#) method), 25
[ls\(\)](#) ([rfslib.ftp_pconnection.FtpPConnection](#) method), 13
[ls\(\)](#) ([rfslib.sftp_pconnection.SftpPConnection](#) method), 9
[ls\(\)](#) ([rfslib.smb12_pconnection.Smb12PConnection](#) method), 17
[ls\(\)](#) ([rfslib.smb23_pconnection.Smb23PConnection](#) method), 21

M

`mkdir()` (*rfslib.abstract_pconnection.PConnection method*), 5
`mkdir()` (*rfslib.fs_pconnection.FsPConnection method*), 25
`mkdir()` (*rfslib.ftp_pconnection.FtpPConnection method*), 13
`mkdir()` (*rfslib.sftp_pconnection.SftpPConnection method*), 9
`mkdir()` (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
`mkdir()` (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
`module`
 rfslib.abstract_pconnection, 3
 rfslib.fs_pconnection, 23
 rfslib.ftp_pconnection, 11
 rfslib.path_utils, 27
 rfslib.sftp_pconnection, 7
 rfslib.smb12_pconnection, 15
 rfslib.smb23_pconnection, 19
`mv()` (*rfslib.abstract_pconnection.PConnection method*), 5
`mv()` (*rfslib.fs_pconnection.FsPConnection method*), 25
`mv()` (*rfslib.ftp_pconnection.FtpPConnection method*), 13
`mv()` (*rfslib.sftp_pconnection.SftpPConnection method*), 9
`mv()` (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
`mv()` (*rfslib.smb23_pconnection.Smb23PConnection method*), 21

P

`path_normalize()` (*in module rfslib.path_utils*), 27
`PConnection` (*class in rfslib.abstract_pconnection*), 3
`pmkdir()` (*rfslib.abstract_pconnection.PConnection method*), 5
`pmkdir()` (*rfslib.fs_pconnection.FsPConnection method*), 25
`pmkdir()` (*rfslib.ftp_pconnection.FtpPConnection method*), 13
`pmkdir()` (*rfslib.sftp_pconnection.SftpPConnection method*), 9
`pmkdir()` (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
`pmkdir()` (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
`pull()` (*rfslib.abstract_pconnection.PConnection method*), 5
`pull()` (*rfslib.fs_pconnection.FsPConnection method*), 25
`pull()` (*rfslib.ftp_pconnection.FtpPConnection method*), 13

`pull()` (*rfslib.sftp_pconnection.SftpPConnection method*), 9
`pull()` (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
`pull()` (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
`push()` (*rfslib.abstract_pconnection.PConnection method*), 5
`push()` (*rfslib.fs_pconnection.FsPConnection method*), 25
`push()` (*rfslib.ftp_pconnection.FtpPConnection method*), 13
`push()` (*rfslib.sftp_pconnection.SftpPConnection method*), 9
`push()` (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
`push()` (*rfslib.smb23_pconnection.Smb23PConnection method*), 21

R

`remove_r_prefix()` (*in module rfslib.path_utils*), 27
`rename()` (*rfslib.abstract_pconnection.PConnection method*), 5
`rename()` (*rfslib.fs_pconnection.FsPConnection method*), 25
`rename()` (*rfslib.ftp_pconnection.FtpPConnection method*), 13
`rename()` (*rfslib.sftp_pconnection.SftpPConnection method*), 9
`rename()` (*rfslib.smb12_pconnection.Smb12PConnection method*), 17
`rename()` (*rfslib.smb23_pconnection.Smb23PConnection method*), 21
`rfslib.abstract_pconnection`
 module, 3
`rfslib.fs_pconnection`
 module, 23
`rfslib.ftp_pconnection`
 module, 11
`rfslib.path_utils`
 module, 27
`rfslib.sftp_pconnection`
 module, 7
`rfslib.smb12_pconnection`
 module, 15
`rfslib.smb23_pconnection`
 module, 19
`rm()` (*rfslib.abstract_pconnection.PConnection method*), 5
`rm()` (*rfslib.fs_pconnection.FsPConnection method*), 25
`rm()` (*rfslib.ftp_pconnection.FtpPConnection method*), 13
`rm()` (*rfslib.sftp_pconnection.SftpPConnection method*), 9

| | | | |
|----------------------|---|----------------------|---|
| <code>rm()</code> | <code>(rfslib.smb12_pconnection.Smb12PConnection method), 17</code> | <code>touch()</code> | <code>(rfslib.ftp_pconnection.FtpPConnection method), 13</code> |
| <code>rm()</code> | <code>(rfslib.smb23_pconnection.Smb23PConnection method), 21</code> | <code>touch()</code> | <code>(rfslib.sftp_pconnection.SftpPConnection method), 9</code> |
| <code>rmdir()</code> | <code>(rfslib.abstract_pconnection.PConnection method), 5</code> | <code>touch()</code> | <code>(rfslib.smb12_pconnection.Smb12PConnection method), 17</code> |
| <code>rmdir()</code> | <code>(rfslib.fs_pconnection.FsPConnection method), 25</code> | <code>touch()</code> | <code>(rfslib.smb23_pconnection.Smb23PConnection method), 21</code> |
| <code>rmdir()</code> | <code>(rfslib.ftp_pconnection.FtpPConnection method), 13</code> | | |
| <code>rmdir()</code> | <code>(rfslib.sftp_pconnection.SftpPConnection method), 9</code> | | |
| <code>rmdir()</code> | <code>(rfslib.smb12_pconnection.Smb12PConnection method), 17</code> | | |
| <code>rmdir()</code> | <code>(rfslib.smb23_pconnection.Smb23PConnection method), 21</code> | | |
| <code>rpull()</code> | <code>(rfslib.abstract_pconnection.PConnection method), 5</code> | | |
| <code>rpull()</code> | <code>(rfslib.fs_pconnection.FsPConnection method), 25</code> | | |
| <code>rpull()</code> | <code>(rfslib.ftp_pconnection.FtpPConnection method), 13</code> | | |
| <code>rpull()</code> | <code>(rfslib.sftp_pconnection.SftpPConnection method), 9</code> | | |
| <code>rpull()</code> | <code>(rfslib.smb12_pconnection.Smb12PConnection method), 17</code> | | |
| <code>rpull()</code> | <code>(rfslib.smb23_pconnection.Smb23PConnection method), 21</code> | | |
| <code>rpush()</code> | <code>(rfslib.abstract_pconnection.PConnection method), 5</code> | | |
| <code>rpush()</code> | <code>(rfslib.fs_pconnection.FsPConnection method), 25</code> | | |
| <code>rpush()</code> | <code>(rfslib.ftp_pconnection.FtpPConnection method), 13</code> | | |
| <code>rpush()</code> | <code>(rfslib.sftp_pconnection.SftpPConnection method), 9</code> | | |
| <code>rpush()</code> | <code>(rfslib.smb12_pconnection.Smb12PConnection method), 17</code> | | |
| <code>rpush()</code> | <code>(rfslib.smb23_pconnection.Smb23PConnection method), 21</code> | | |

U

| | |
|-----------------------|---|
| <code>unlink()</code> | <code>(rfslib.abstract_pconnection.PConnection method), 5</code> |
| <code>unlink()</code> | <code>(rfslib.fs_pconnection.FsPConnection method), 25</code> |
| <code>unlink()</code> | <code>(rfslib.ftp_pconnection.FtpPConnection method), 13</code> |
| <code>unlink()</code> | <code>(rfslib.sftp_pconnection.SftpPConnection method), 9</code> |
| <code>unlink()</code> | <code>(rfslib.smb12_pconnection.Smb12PConnection method), 17</code> |
| <code>unlink()</code> | <code>(rfslib.smb23_pconnection.Smb23PConnection method), 21</code> |

X

| | |
|--------------------|---|
| <code>xls()</code> | <code>(rfslib.abstract_pconnection.PConnection method), 5</code> |
| <code>xls()</code> | <code>(rfslib.fs_pconnection.FsPConnection method), 25</code> |
| <code>xls()</code> | <code>(rfslib.ftp_pconnection.FtpPConnection method), 13</code> |
| <code>xls()</code> | <code>(rfslib.sftp_pconnection.SftpPConnection method), 9</code> |
| <code>xls()</code> | <code>(rfslib.smb12_pconnection.Smb12PConnection method), 17</code> |
| <code>xls()</code> | <code>(rfslib.smb23_pconnection.Smb23PConnection method), 21</code> |

S

`SftpPConnection` (class in `rfslib.sftp_pconnection`), 7

`Smb12PConnection` (class in `rfslib.smb12_pconnection`), 15

`Smb23PConnection` (class in `rfslib.smb23_pconnection`), 19

T

| | |
|----------------------|--|
| <code>touch()</code> | <code>(rfslib.abstract_pconnection.PConnection method), 5</code> |
| <code>touch()</code> | <code>(rfslib.fs_pconnection.FsPConnection method), 25</code> |