python-unrar Documentation

Release 0.3

Matias Bordese

Aug 24, 2017

Contents

1 `rarfile` — Work with RAR archives **3**
- 1.1 RarFile Objects . . . 3
- 1.2 RarInfo Objects . . . 4

2 `unrarlib` — ctypes wrapper for UnRAR library **7**
- 2.1 RAROpenArchiveDataEx . . . 10
- 2.2 RARHeaderDataEx . . . 11

3 `constants` — constants used by the UnRAR library **15**

4 Indices and tables **17**

Python Module Index **19**

python-unrar provides a high-level module for reading and listing RAR archives inspired on Python's zipfile.

rarfile is built on top of *unrarlib*, a low-level ctypes wrapper for the UnRAR library.

Contents:

CHAPTER 1

`rarfile` — Work with RAR archives

The RAR file format is a common archive and compression standard. This module provides tools to read and list a RAR file.

This module is based on the UnRAR library (provided by RARLAB) through a ctypes wrapper, and inspired on Python's ZipFile.

The module defines the following items:

exception `rarfile.`**`BadRarFile`**
The error raised for bad RAR files.

class `rarfile.`**`RarFile`**
The class for reading RAR files. See section *RarFile Objects* for constructor details.

class `rarfile.`**`RarInfo`**
Class used to represent information about a member of an archive. Instances of this class are returned by the `getinfo()` and `infolist()` methods of *`RarFile`* objects. Most users of the *`rarfile`* module will not need to create these, but only use those created by this module. *header* should be a RARHeaderDataEx instance as returned by *`unrarlib`*; the fields are described in section *RarInfo Objects*.

`rarfile.`**`is_rarfile`**(*filename*)
Returns `True` if *filename* is a valid RAR file based on its magic number, otherwise returns `False`.

See also:

RARLAB Official RAR site.

RAR addons RAR addons where you can download UnRAR library sources.

RarFile Objects

class `rarfile.`**`RarFile`**(*file*[, *mode='r'*])
Open a RAR file, where *file* should be a path to a file (a string). The *mode* parameter should be `'r'` to read an existing file (only allowed mode at the moment).

`RarFile.`**`getinfo`**(*name*)
Return a *RarInfo* object with information about the archive member *name*. Calling *getinfo()* for a name not currently contained in the archive will raise a `KeyError`.

`RarFile.`**`infolist`**()
Return a list containing a *RarInfo* object for each member of the archive. The objects are in the same order as their entries in the actual RAR file on disk if an existing archive was opened.

`RarFile.`**`namelist`**()
Return a list of archive members by name.

`RarFile.`**`open`**(*member*[, *pwd*])
Extract a member from the archive as a file-like object (see Python's `io.BytesIO`). *member* is the name of the file in the archive, or a *RarInfo* object. *pwd* is the password used for encrypted files.

New in version 0.3.

`RarFile.`**`read`**(*name*[, *pwd*])
Return the bytes of the file *member* in the archive. *member* is the name of the file in the archive, or a *RarInfo* object. *pwd* is the password used for encrypted files and, if specified, it will override the default password set with *setpassword()*.

New in version 0.3.

`RarFile.`**`extract`**(*member*, *path=None*, *pwd=None*)
Extract a member from the archive to the current working directory; *member* must be its full name or a *RarInfo* object). Its file information is extracted as accurately as possible. *path* specifies a different directory to extract to. *member* can be a filename or a *RarInfo* object. *pwd* is the password used for encrypted files.

`RarFile.`**`extractall`**(*path=None*, *members=None*, *pwd=None*)
Extract all members from the archive to the current working directory. *path* specifies a different directory to extract to. *members* is optional and must be a subset of the list returned by *namelist()*. *pwd* is the password used for encrypted files.

> **Warning:** Never extract archives from untrusted sources without prior inspection. It is possible that files are created outside of *path*, e.g. members that have absolute filenames starting with `"/"` or filenames with two dots `".."`.

`RarFile.`**`printdir`**()
Print a table of contents for the archive to `sys.stdout`.

`RarFile.`**`setpassword`**(*pwd*)
Set *pwd* as default password to extract encrypted files.

`RarFile.`**`testrar`**()
Read all the files in the archive and check their CRC's and file headers. Return the name of the first bad file, or else return `None`.

The following data attribute is also available:

`RarFile.`**`comment`**
The comment text associated with the RAR file, if any.

RarInfo Objects

Instances of the *RarInfo* class are returned by the `getinfo()` and `infolist()` methods of *RarFile* objects. Each object stores information about a single member of the RAR archive.

Instances have the following attributes:

RarInfo.**filename**
Name of the file in the archive.

RarInfo.**date_time**
The time and date of the last modification to the archive member. This is a tuple of six values:

Index	Value
0	Year (>= 1980)
1	Month (one-based)
2	Day of month (one-based)
3	Hours (zero-based)
4	Minutes (zero-based)
5	Seconds (zero-based)

Note: The RAR file format does not support timestamps before 1980.

RarInfo.**compress_type**
Type of compression for the archive member.

RarInfo.**comment**
Comment for the individual archive member.

RarInfo.**create_system**
System which created RAR archive.

RarInfo.**extract_version**
RAR version needed to extract archive.

RarInfo.**flag_bits**
RAR flag bits.

RarInfo.**CRC**
CRC-32 of the uncompressed file.

RarInfo.**compress_size**
Size of the compressed data.

RarInfo.**file_size**
Size of the uncompressed file.

CHAPTER 2

unrarlib — ctypes wrapper for UnRAR library

The RAR file format is a common archive and compression standard. This module provides a ctypes wrapper for UnRAR library (provided by RARLAB).

Most of this information is extracted from RAR library documentation.

The module defines the following items:

exception unrarlib.**UnrarException**
The error raised for possible library errors.

class unrarlib.**RAROpenArchiveDataEx**
Class mapping the library RAROpenArchiveDataEx C struct:

```
struct RAROpenArchiveData{
    char         *ArcName;
    wchar_t      *ArcNameW;
    unsigned int OpenMode;
    unsigned int OpenResult;
    char         *CmtBuf;
    unsigned int CmtBufSize;
    unsigned int CmtSize;
    unsigned int CmtState;
    unsigned int Flags;
    unsigned int Reserved[32];
};
```

Using the name fields described above you can access the respective values through an instance as they were instance attributes.

See section *RAROpenArchiveDataEx* for details.

class unrarlib.**RARHeaderDataEx**
Class mapping the library RARHeaderDataEx C struct:

```
struct RARHeaderDataEx{
    char         ArcName[1024];
    wchar_t      ArcNameW[1024];
```

```
    char         FileName[1024];
    wchar_t      FileNameW[1024];
    unsigned int Flags;
    unsigned int PackSize;
    unsigned int PackSizeHigh;
    unsigned int UnpSize;
    unsigned int UnpSizeHigh;
    unsigned int HostOS;
    unsigned int FileCRC;
    unsigned int FileTime;
    unsigned int UnpVer;
    unsigned int Method;
    unsigned int FileAttr;
    char         *CmtBuf;
    unsigned int CmtBufSize;
    unsigned int CmtSize;
    unsigned int CmtState;
    unsigned int Reserved[1024];
};
```

See section *RARHeaderDataEx* for details.

unrarlib.**dostime_to_timetuple**(*dostime*)
Convert an MS-DOS format date time to a Python time tuple.

unrarlib.**RAROpenArchiveEx**(*archive_data*)
Open RAR archive and allocate memory structures. *archive_data* should be a pointer to an initialized *RAROpenArchiveDataEx*. When it succeeds, it loads the archive information into *archive_data* and returns a handle identifying the open archive, to be used as argument to the other functions in the module. In case of error, it raises an UnrarException (you can confirm which the error was by checking *RAROpenArchiveDataEx.OpenResult*).

unrarlib.**RARCloseArchive**(*handle*)
Close RAR archive and release allocated memory. It must be called when archive processing is finished, even if the archive processing was stopped due to an error.

unrarlib.**RARReadHeaderEx**(*handle*, *header_data*)
Read header of file in archive. *header_data* should be a pointer to an initialized *RARHeaderDataEx*, that would get filled with the member details.

unrarlib.**RARProcessFile**(*handle*, *operation*, *dest_path*, *dest_name*)
Performs action and moves the current position in the archive to the next file. Extract or test the current file from the archive opened in *constants*.RAR_OM_EXTRACT mode. If the mode *constants*.RAR_OM_LIST is set, then a call to this function will simply skip the archive position to the next file.

Possible operations are:

***constants*.RAR_SKIP** Move to the next file in the archive. If the archive is solid and *constants*.RAR_OM_EXTRACT mode was set when the archive was opened, the current file will be processed - the operation will be performed slower than a simple seek.

***constants*.RAR_TEST** Test the current file and move to the next file in the archive. If the archive was opened with *constants*.RAR_OM_LIST mode, the operation is equal to *constants*.RAR_SKIP.

***constants*.RAR_EXTRACT** Extract the current file and move to the next file. If the archive was opened with *constants*.RAR_OM_LIST mode, the operation is equal to *constants*.RAR_SKIP.

unrarlib.**RARSetPassword**(*handle*, *pwd*)
Set a password to decrypt files when processing.

unrarlib.**RARSetCallback**(*handle*, *callback*, *user_data*)
Set a user-defined callback function to process unrar events.

callback is a user-defined function, taking four parameters:

msg Type of event.

user_data User data passed to callback function.

P1, P2 Event dependent parameters.

You need to wrap your *callback* function to use the right calling convention; for that you can use *unrarlib*.UNRARCALLBACK function, that will set the correct argument types and return value for your OS.

Possible events:

***constants*.UCM_CHANGEVOLUME**

Process volume change.

P1 Points to the zero terminated name of the next volume.

P2

The function call mode:

***constants*.RAR_VOL_ASK** Required volume is absent. The function should prompt user and return a positive value to retry or return -1 value to terminate operation. The function may also specify a new volume name, placing it to the address specified by P1 parameter.

***constants*.RAR_VOL_NOTIFY** Required volume is successfully opened. This is a notification call and volume name modification is not allowed. The function should return a positive value to continue or -1 to terminate operation.

***constants*.UCM_PROCESSDATA** Process unpacked data. It may be used to read a file while it is being extracted or tested without actual extracting file to disk. Return a positive value to continue process or -1 to cancel the archive operation

P1 Address pointing to the unpacked data. Function may refer to the data but must not change it.

P2 Size of the unpacked data. It is guaranteed only that the size will not exceed the maximum dictionary size (4 Mb in RAR 3.0).

***constants*.UCM_NEEDPASSWORD** DLL needs a password to process archive. This message must be processed if you wish to be able to handle archives with encrypted file names. It can be also used as replacement of RARSetPassword function even for usual encrypted files with non-encrypted names.

P1 Address pointing to the buffer for a password. You need to copy a password here.

P2 Size of the password buffer.

Other functions of unrarlib should not be called from the callback function. There is no return value. For an example of using callbacks, you can check *rarfile.RarFile.open()* implementation.

unrarlib.**RARGetDllVersion**()
Return library API version.

RAROpenArchiveDataEx

class unrarlib.**RAROpenArchiveDataEx** (*filename*[, *mode=constants.RAR_OM_LIST*])
Initialize a RAROpenArchiveDataEx instance to open *filename* using the indicated mode. *mode* should be one of the possible open modes defined in *constants*.

RAROpenArchiveDataEx.**ArcName**
Input parameter, a string containing the archive name.

RAROpenArchiveDataEx.**ArcNameW**
Input parameter, a Unicode string containing the archive name or None if Unicode name is not specified.

RAROpenArchiveDataEx.**OpenMode**
Input parameter.

Possible values:

***constants*.RAR_OM_LIST** Open archive for reading file headers only.

***constants*.RAR_OM_EXTRACT** Open archive for testing and extracting files.

***constants*.RAR_OM_LIST_INCSPLIT** Open archive for reading file headers only. If you open an archive in such mode, RARReadHeaderEx will return all file headers, including those with "file continued from previous volume" flag. In case of *constants*.RAR_OM_LIST such headers are automatically skipped. So if you process RAR volumes in *constants*.RAR_OM_LIST_INCSPLIT mode, you will get several file header records for same file if file is split between volumes. For such files only the last file header record will contain the correct file CRC and if you wish to get the correct packed size, you need to sum up packed sizes of all parts.

RAROpenArchiveDataEx.**OpenResult**
Output parameter.

Possible values:

***constants*.SUCCESS** Success

***constants*.ERAR_NO_MEMORY** Not enough memory to initialize data structures

***constants*.ERAR_BAD_DATA** Archive header broken

***constants*.ERAR_BAD_ARCHIVE** File is not valid RAR archive

***constants*.ERAR_UNKNOWN_FORMAT** Unknown encryption used for archive headers

***constants*.ERAR_EOPEN** File open error

RAROpenArchiveDataEx.**CmtBuf**
Buffer for archive comments. Maximum comment size is limited to 64Kb. If the comment text is larger than the buffer size, the comment text will be truncated.

RAROpenArchiveDataEx.**CmtBufSize**
Input parameter which should contain size of buffer for archive comments.

RAROpenArchiveDataEx.**CmtSize**
Output parameter containing size of comments actually read into the buffer, cannot exceed *CmtBufSize*.

RAROpenArchiveDataEx.**CmtState**
Output parameter.

Possible values:

***constants*.RAR_NO_COMMENTS** Comments not present

***constants*.RAR_COMMENTS_SUCCESS** Comments read completely

constants.**ERAR_NO_MEMORY** Not enough memory to extract comments

constants.**ERAR_BAD_DATA** Broken comment

constants.**ERAR_UNKNOWN_FORMAT** Unknown comment format

constants.**ERAR_SMALL_BUF** Buffer too small, comments not completely read

RAROpenArchiveDataEx.**Flags**
Output parameter. Combination of bit flags.

Possible values:

0x0001 - Volume attribute (archive volume)
0x0002 - Archive comment present
0x0004 - Archive lock attribute
0x0008 - Solid attribute (solid archive)
0x0010 - New volume naming scheme ('volname.partN.rar')
0x0020 - Authenticity information present
0x0040 - Recovery record present
0x0080 - Block headers are encrypted
0x0100 - First volume (set only by RAR 3.0 and later)

RAROpenArchiveDataEx.**Reserved**
Reserved for future use. Must be zero.

RARHeaderDataEx

class unrarlib.**RARHeaderDataEx**()
Initialize an empty RARHeaderDataEx instance, to be populated with the details returned by *RARReadHeaderEx()*.

RARHeaderDataEx.**ArcName**
Output parameter which contains a zero terminated string of the current archive name. May be used to determine the current volume name.

RARHeaderDataEx.**FileName**
Output parameter which contains a zero terminated string of the file name in OEM (DOS) encoding.

RARHeaderDataEx.**Flags**
Output parameter which contains file flags:

0x01 - file continued from previous volume
0x02 - file continued on next volume
0x04 - file encrypted with password
0x08 - file comment present
0x10 - compression of previous files is used (solid flag)

[bits 7 6 5]
0 0 0 - dictionary size 64 Kb
0 0 1 - dictionary size 128 Kb
0 1 0 - dictionary size 256 Kb
0 1 1 - dictionary size 512 Kb
1 0 0 - dictionary size 1024 Kb

1 0 1 - dictionary size 2048 KB
1 1 0 - dictionary size 4096 KB
1 1 1 - file is directory

Other bits are reserved.

RARHeaderDataEx.**PackSize**
Output parameter means packed file size or size of the file part if file was split between volumes.

RARHeaderDataEx.**UnpSize**
Output parameter - unpacked file size.

RARHeaderDataEx.**HostOS**
Output parameter - operating system used for archiving:

constants.RAR_DOS
constants.RAR_OS2
constants.RAR_WIN
constants.RAR_UNIX

RARHeaderDataEx.**FileCRC**
Output parameter which contains unpacked file CRC. In case of file parts split between volumes only the last part contains the correct CRC and it is accessible only in *constants*.RAR_OM_LIST_INCSPLIT listing mode.

RARHeaderDataEx.**FileTime**
Output parameter - contains date and time in standard MS DOS format.

RARHeaderDataEx.**UnpVer**
Output parameter - RAR version needed to extract file. It is encoded as 10 * Major version + minor version.

RARHeaderDataEx.**Method**
Output parameter - packing method.

RARHeaderDataEx.**FileAttr**
Output parameter - file attributes.

RARHeaderDataEx.**CmtBuf**
File comments support is not implemented in the new DLL version yet. Now *CmtState* is always *constants*.RAR_NO_COMMENTS.

RARHeaderDataEx.**CmtBufSize**
Input parameter which should contain size of buffer for archive comments.

RARHeaderDataEx.**CmtSize**
Output parameter containing size of comments actually read into the buffer, should not exceed *CmtBufSize*.

RARHeaderDataEx.**CmtState**
Output parameter.

Possible values:

***constants*.RAR_NO_COMMENTS** Absent comments

***constants*.RAR_COMMENTS_SUCCESS** Comments read completely

***constants*.ERAR_NO_MEMORY** Not enough memory to extract comments

***constants*.ERAR_BAD_DATA** Broken comment

***constants*.ERAR_UNKNOWN_FORMAT** Unknown comment format

***constants*.ERAR_SMALL_BUF** Buffer too small, comments not completely read

See also:

UnRAR Manual `UnRAR library manual`

RARLAB Official RAR site.

RAR addons RAR addons where you can download UnRAR library sources. Check source files to get more detailed information.

CHAPTER 3

constants — constants used by the UnRAR library

General

constants.SUCCESS = 0

Open Modes

constants.RAR_OM_LIST = 0
constants.RAR_OM_EXTRACT = 1
constants.RAR_OM_LIST_INCSPLIT = 2

Processing operations

constants.RAR_SKIP = 0
constants.RAR_TEST = 1
constants.RAR_EXTRACT = 2
constants.RAR_CANCEL_EXTRACT = -1

Errors

constants.ERAR_END_ARCHIVE = 10
constants.ERAR_NO_MEMORY = 11
constants.ERAR_BAD_DATA = 12
constants.ERAR_BAD_ARCHIVE = 13
constants.ERAR_UNKNOWN_FORMAT = 14

constants.ERAR_EOPEN = 15
constants.ERAR_ECREATE = 16
constants.ERAR_ECLOSE = 17
constants.ERAR_EREAD = 18
constants.ERAR_EWRITE = 19
constants.ERAR_SMALL_BUF = 20
constants.ERAR_UNKNOWN = 21
constants.ERAR_MISSING_PASSWORD = 22

Comments

constants.RAR_NO_COMMENTS = 0
constants.RAR_COMMENTS_SUCCESS = 1

Host OS

constants.RAR_DOS = 0
constants.RAR_OS2 = 1
constants.RAR_WIN = 2
constants.RAR_UNIX = 3

Callback messages

constants.UCM_CHANGEVOLUME = 0
constants.UCM_PROCESSDATA = 1
constants.UCM_NEEDPASSWORD = 2
constants.UCM_CHANGEVOLUMEW = 3
constants.UCM_NEEDPASSWORDW = 4

Change volume callback's messages

constants.RAR_VOL_ASK = 0
constants.RAR_VOL_NOTIFY = 1

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

c

constants, 15

r

rarfile, 3

u

unrarlib, 7

Index

A

ArcName (unrarlib.RARHeaderDataEx attribute), 11
ArcName (unrarlib.RAROpenArchiveDataEx attribute), 10
ArcNameW (unrarlib.RAROpenArchiveDataEx attribute), 10

B

BadRarFile, 3

C

CmtBuf (unrarlib.RARHeaderDataEx attribute), 12
CmtBuf (unrarlib.RAROpenArchiveDataEx attribute), 10
CmtBufSize (unrarlib.RARHeaderDataEx attribute), 12
CmtBufSize (unrarlib.RAROpenArchiveDataEx attribute), 10
CmtSize (unrarlib.RARHeaderDataEx attribute), 12
CmtSize (unrarlib.RAROpenArchiveDataEx attribute), 10
CmtState (unrarlib.RARHeaderDataEx attribute), 12
CmtState (unrarlib.RAROpenArchiveDataEx attribute), 10
comment (rarfile.RarFile attribute), 4
comment (rarfile.RarInfo attribute), 5
compress_size (rarfile.RarInfo attribute), 5
compress_type (rarfile.RarInfo attribute), 5
constants (module), 15
CRC (rarfile.RarInfo attribute), 5
create_system (rarfile.RarInfo attribute), 5

D

date_time (rarfile.RarInfo attribute), 5
dostime_to_timetuple() (in module unrarlib), 8

E

extract() (rarfile.RarFile method), 4
extract_version (rarfile.RarInfo attribute), 5
extractall() (rarfile.RarFile method), 4

F

file_size (rarfile.RarInfo attribute), 5
FileAttr (unrarlib.RARHeaderDataEx attribute), 12
FileCRC (unrarlib.RARHeaderDataEx attribute), 12
filename (rarfile.RarInfo attribute), 5
FileName (unrarlib.RARHeaderDataEx attribute), 11
FileTime (unrarlib.RARHeaderDataEx attribute), 12
flag_bits (rarfile.RarInfo attribute), 5
Flags (unrarlib.RARHeaderDataEx attribute), 11
Flags (unrarlib.RAROpenArchiveDataEx attribute), 11

G

getinfo() (rarfile.RarFile method), 3

H

HostOS (unrarlib.RARHeaderDataEx attribute), 12

I

infolist() (rarfile.RarFile method), 4
is_rarfile() (in module rarfile), 3

M

Method (unrarlib.RARHeaderDataEx attribute), 12

N

namelist() (rarfile.RarFile method), 4

O

open() (rarfile.RarFile method), 4
OpenMode (unrarlib.RAROpenArchiveDataEx attribute), 10
OpenResult (unrarlib.RAROpenArchiveDataEx attribute), 10

P

PackSize (unrarlib.RARHeaderDataEx attribute), 12
printdir() (rarfile.RarFile method), 4

R

RARCloseArchive() (in module unrarlib), 8
RarFile (class in rarfile), 3
rarfile (module), 3
RARGetDllVersion() (in module unrarlib), 9
RARHeaderDataEx (class in unrarlib), 7, 11
RarInfo (class in rarfile), 3
RAROpenArchiveDataEx (class in unrarlib), 7, 10
RAROpenArchiveEx() (in module unrarlib), 8
RARProcessFile() (in module unrarlib), 8
RARReadHeaderEx() (in module unrarlib), 8
RARSetCallback() (in module unrarlib), 9
RARSetPassword() (in module unrarlib), 8
read() (rarfile.RarFile method), 4
Reserved (unrarlib.RAROpenArchiveDataEx attribute), 11

S

setpassword() (rarfile.RarFile method), 4

T

testrar() (rarfile.RarFile method), 4

U

UnpSize (unrarlib.RARHeaderDataEx attribute), 12
UnpVer (unrarlib.RARHeaderDataEx attribute), 12
UnrarException, 7
unrarlib (module), 7