py7zr Documentation

Release 0.20.8

Hiroshi Miura

Nov 13, 2023

CONTENTS

CHAPTER

ONE

USER GUIDE

The 7z file format is a popular archive and compression format in recent days. This module provides tools to read, write and list 7z file. Features is not implemented to update and append a 7z file. py7zr does not support self-extracting archive, aka. SFX file, and only support plain 7z archive file.

1.1 Getting started

1.1.1 Install

The py7zr is written by Python and can be downloaded from PyPI(aka. Python Package Index) using standard 'pip' command as like follows;

```
$ pip install py7zr
```

The py7zr depends on several external libraries. You should install these libraries with py7zr. There are PyCryptodomex, PyZstd, PyPPMd, pybcj, texttable, and multivolumefile. There are also dependency whether Brotli or BrotliCFFI depends on your python flavour. These packages are automatically installed when installing with `pip` command.

1.1.2 Dependencies

There are several dependencies to support algorithms and CLI expressions.

Package	Purpose
PyCryptodomex	7zAES encryption
PyZstd	ZStandard compression
PyPPMd	PPMd compression
Brotli	Brotli compression (CPython)
BrotliCFFI	Brotli compression (PyPy)
zipfile-deflate64	DEFLATE64 decompression
pybcj	BCJ filters
multivolumefile	Multi-volume archive read/write
texttable	CLI formatter

Note: There is known issue when you run `py7zr` on Windows platform. issue#527 report that when installed on Azure VM Windows, it failed to import Brotli library even when `pip install` was successfully executed. It is because Brotli

library depends `vc_redist.x64` Microsoft system library which should be come with python distribution, but Brotli library does not find the DLL in search path on certain environment. Please check details at Brotli Issue#782

Current `py7zr` detect the import failure, and raise exception only when user try to compress/decompress with Brotli compression algorithm.

1.1.3 Run Command

'py7zr' is a command script. You can run extracting a target file target.7z then command line become as such as follows;

```
$ py7zr x target.7z
```

When you want to create an archive from a files and directory under the current directory 'd', command line become as such as follows;

```
$ py7zr c target.7z  d/
```

1.2 Command-Line Interfaces

The *py7zr* module provides a simple command-line interface to interact with 7z archives.

If you want to extract a 7z archive into the specified directory, use the *x* subcommand:

```
$ python -m py7zr x monty.7z target-dir/
$ py7zr x monty.7z
```

For a list of the files in a 7z archive, use the *l* subcommand:

```
$ python -m py7zr l monty.7z
$ py7zr l monty.7z
```

1.2.1 Command-line options

l <7z file>
: List files in a 7z file.

x <7z file> [<output_dir>]
: Extract 7z file into target directory.

c <7z file> <base_dir>
: Create 7zip archive from base_directory

a <7z file> <base_dir>
: Append files from base_dir to existent 7zip archive.

i <7z file>
: Show archive information of specified 7zip archive.

t <7z file>
: Test whether the 7z file is valid or not.

1.2.2 Common command options

-P --password
: Extract, list or create password protected archive. py7zr will prompt user input.

--verbose
: Show verbose debug log.

1.2.3 Create command options

-v | --volume {Size}[b|k|m|g]
: Create multi-volume archive with Size. Usable with 'c' sub-command.

1.3 Programming APIs

1.3.1 Extraction

Here is a several example for extraction from your python program. You can write it with very clean syntax because py7zr supports context manager.

```
import py7zr
with py7zr.SevenZipFile("Archive.7z", 'r') as archive:
    archive.extractall(path="/tmp")
```

This example extract a 7-zip archive file "Archive.7z" into "/tmp" target directory.

1.3.2 Make archive

Here is a simple example to make 7-zip archive.

```
import py7zr
with py7zr.SevenZipFile("Archive.7z", 'w') as archive:
    archive.writeall("target/")
```

1.3.3 Append files to archive

Here is a simple example to append some files into existent 7-zip archive.

```
import py7zr
with py7zr.SevenZipFile("Archive.7z", 'a') as archive:
    archive.write("additional_file.txt")
```

1.3.4 Extraction from multi-volume archive

You should concatenate multi-volume archives into single archive file before call py7zr, or consider using files wrapping class that handle multiple files as a virtual single file, (ex. multivolumefile library)

```
import py7zr
filenames = ['example.7z.0001', 'example.7z.0002']
with open('result.7z', 'ab') as outfile:  # append in binary mode
    for fname in filenames:
        with open(fname, 'rb') as infile:        # open in binary mode also
            outfile.write(infile.read())
with py7zr.SevenZipFile("result.7z", "r") as archive:
    archive.extractall()
os.unlink("result.7z")
```

Here is another example. This example use multivolumefile library. The multivolumefile library is in pre-alpha status, so it is not recommend to use production system.

```
pip install py7zr multivolumefile
```

When there are files named, 'example.7z.0001', 'example.7z.0002', and so on, following code will extract multi-volume archive.

```
import multivolumefile
import py7zr
with multivolumefile.open('example.7z', mode='rb') as target_archive:
    with SevenZipFile(target_archive, 'r') as archive:
        archive.extractall()
```

If you want to create multi volume archive using multivolumefile library, following example do it for you.

```
import multivolumefile
import py7zr

target = pathlib.Path('/target/directory/')
with multivolumefile.open('example.7z', mode='wb', volume_size=10240) as target_archive:
    with SevenZipFile(target_archive, 'w') as archive:
        archive.writeall(target, 'target')
```

1.4 Presentation material

See `Introductory presentation(PDF)`, and `Introductory presentation(ODP)`.

CHAPTER

TWO

API DOCUMENTATION

2.1 py7zr — 7-Zip archive library

The module is built upon awesome development effort and knowledge of *pylzma* module and its *py7zlib.py* program by Joachim Bauch. Great appreciation for Joachim!

The module defines the following items:

exception py7zr.**Bad7zFile**

The error raised for bad 7z files.

class py7zr.**SevenZipFile**

The class for reading 7z files. See section *sevenzipfile-object*

class py7zr.**FileInfo**

The class used to represent information about a member of an archive file. See section

py7zr.**is_7zfile**(*filename*)

Returns `True` if *filename* is a valid 7z file based on its magic number, otherwise returns `False`. *filename* may be a file or file-like object too.

py7zr.**unpack_7zarchive**(*archive*, *path*, *extra=None*)

Helper function to intend to use with `shutil` module which offers a number of high-level operations on files and collections of files. Since `shutil` has a function to register decompressor of archive, you can register an helper function and then you can extract archive by calling `shutil.unpack_archive()`

```
shutil.register_unpack_format('7zip', ['.7z'], unpack_7zarchive)
shutil.unpack_archive(filename, [, extract_dir])
```

py7zr.**pack_7zarchive**(*archive*, *path*, *extra=None*)

Helper function to intend to use with `shutil` module which offers a number of high-level operations on files and collections of files. Since `shutil` has a function to register maker of archive, you can register an helper function and then you can produce archive by calling `shutil.make_archive()`

```
shutil.register_archive_format('7zip', pack_7zarchive, description='7zip archive')
shutil.make_archive(base_name, '7zip', base_dir)
```

See also:

(external link) shutil `shutil` module offers a number of high-level operations on files and collections of files.

2.2 Class description

2.2.1 ArchiveInfo Object

class py7zr.**ArchiveInfo**(*filename*, *stat*, *header_size*, *method_names*, *solid*, *blocks*, *uncompressed*)

Data only python object to hold information of archive. The object can be retrieved by *archiveinfo()* method of *SevenZipFile* object.

py7zr.**filename: str**

filename of 7zip archive. If SevenZipFile object is created from BinaryIO object, it becomes None.

py7zr.**stat: stat_result**

fstat object of 7zip archive. If SevenZipFile object is created from BinaryIO object, it becomes None.

py7zr.**header_size: int**

header size of 7zip archive.

py7zr.**method_names: List[str]**

list of method names used in 7zip archive. If method is not supported by py7zr, name has a postfix asterisk(*) mark.

py7zr.**solid: bool**

Whether is 7zip archive a solid compression or not.

py7zr.**blocks: int**

number of compression block(s)

py7zr.**uncompressed: int**

total uncompressed size of files in 7zip archive

2.2.2 SevenZipFile Object

class py7zr.**SevenZipFile**(*file*, *mode='r'*, *filters=None*, *dereference=False*, *password=None*)

Open a 7z file, where *file* can be a path to a file (a string), a file-like object or a *path-like object*.

The *mode* parameter should be `'r'` to read an existing file, `'w'` to truncate and write a new file, `'a'` to append to an existing file, or `'x'` to exclusively create and write a new file. If *mode* is `'x'` and *file* refers to an existing file, a `FileExistsError` will be raised. If *mode* is `'r'` or `'a'`, the file should be seekable.

The *filters* parameter controls the compression algorithms to use when writing files to the archive.

SevenZipFile class has a capability as context manager. It can handle 'with' statement.

If dereference is False, add symbolic and hard links to the archive. If it is True, add the content of the target files to the archive. This has no effect on systems that do not support symbolic links.

When password given, py7zr handles an archive as an encrypted one.

SevenZipFile.**close**()

Close the archive file and release internal buffers. You must call *close()* before exiting your program or most records will not be written.

SevenZipFile.**getnames**()

Return a list of archive files by name.

SevenZipFile.**needs_password**()

Return *True* if the archive is encrypted, or is going to create encrypted archive. Otherwise return *False*

SevenZipFile.**extractall**(*path=None*)

Extract all members from the archive to current working directory. *path* specifies a different directory to extract to.

SevenZipFile.**extract**(*path=None*, *targets=None*)

Extract specified pathspec archived files to current working directory. 'path' specifies a differenct directory to extract to.

'targets' is a list of archived files to be extracted. py7zr looks for files and directories as same as specified in 'targets'.

Once extract() called, the SevenZipFIle object become exhausted and EOF state. If you want to call read(), readall(), extract(), extractall() again, you should call reset() before it.

CAUTION when specifying files and not specifying parent directory, py7zr will fails with no such directory. When you want to extract file 'somedir/somefile' then pass a list: ['somedirectory', 'somedir/somefile'] as a target argument.

Please see 'tests/test_basic.py: test_py7zr_extract_and_getnames()' for example code.

```
filter_pattern = re.compile(r'scripts.*')
with SevenZipFile('archive.7z', 'r') as zip:
     allfiles = zip.getnames()
     targets = [f if filter_pattern.match(f) for f in allfiles]
with SevenZipFile('archive.7z', 'r') as zip:
     zip.extract(targets=targets)
```

SevenZipFile.**readall**()

Extract all members from the archive to memory and returns dictionary object. Returned dictionary has a form of Dict[filename: str, BinaryIO: io.BytesIO object]. Once readall() called, the SevenZipFIle object become exhausted and EOF state. If you want to call read(), readall(), extract(), extractall() again, you should call reset() before it. You can get extracted data from dictionary value as such

```
with SevenZipFile('archive.7z', 'r') as zip:
    for fname, bio in zip.readall().items():
        print(f'{fname}: {bio.read(10)}...')
```

SevenZipFile.**read**(*targets=None*)

Extract specified list of target archived files to dictionary object. 'targets' is a list of archived files to be extracted. py7zr looks for files and directories as same as specified in 'targets'. When targets is None, it behave as same as readall(). Once read() called, the SevenZipFIle object become exhausted and EOF state. If you want to call read(), readall(), extract(), extractall() again, you should call reset() before it.

```
filter_pattern = re.compile(r'scripts.*')
with SevenZipFile('archive.7z', 'r') as zip:
     allfiles = zip.getnames()
     targets = [f for f in allfiles if filter_pattern.match(f)]
with SevenZipFile('archive.7z', 'r') as zip:
     for fname, bio in zip.read(targets).items():
         print(f'{fname}: {bio.read(10)}...')
```

SevenZipFile.**list**()

Return a List[FileInfo].

SevenZipFile.**archiveinfo**()

Return a ArchiveInfo object.

SevenZipFile.**test**()

Read all the archive file and check a packed CRC. Return `True` if CRC check passed, and return `False` when detect defeat, or return `None` when the archive don't have a CRC record.

SevenZipFile.**testzip**()

Read all the files in the archive and check their CRCs. Return the name of the first bad file, or else return `None`. When the archive don't have a CRC record, it return `None`.

SevenZipFile.**write**(*filename*, *arcname=None*)

Write the file named *filename* to the archive, giving it the archive name *arcname* (by default, this will be the same as *filename*, but without a drive letter and with leading path separators removed). The archive must be open with mode `'w'`

SevenZipFile.**writeall**(*filename*, *arcname=None*)

Write the directory and its sub items recursively into the archive, giving the archive name *arcname* (by default, this will be the same as *filename*, but without a drive letter and with leading path seaprator removed).

If you want to store directories and files, putting *arcname* is good idea. When filename is 'C:/a/b/c' and arcname is 'c', with a file exist as 'C:/a/b/c/d.txt', then archive listed as ['c', 'c/d.txt'], the former as directory.

SevenZipFile.**set_encrypted_header**(*mode*)

Set header encryption mode. When encrypt header, set mode to *True*, otherwise *False*. Default is *False*.

SevenZipFile.**set_encoded_header_mode**(*mode*)

Set header encode mode. When encode header data, set mode to *True*, otherwise *False*. Default is *True*.

2.3 Compression Methods

'py7zr' supports algorithms and filters which lzma module and liblzma support. It also support BZip2 and Deflate that are implemented in python core libraries, and ZStandard with third party libraries. *py7zr*, python3 core lzma module and *liblzma* do not support some algorithms such as PPMd, BCJ2 and Deflate64.

Here is a table of algorithms.

#	Category	Algorithm	Note
1	• Compression • Decompression	LZMA2	default (LZMA2+BCJ)
2		LZMA	
3		Bzip2	
4		Deflate	
5		COPY	
6		PPMd	depend on pyppmd
7		ZStandard	depend on pyzstd
8		Brotli	depend on brotli,brotliCFFI
9	• Filter	BCJ	**(X86, ARM, PPC, ARMT, SPARC,** IA64) depend on bcj-cffi
10		Delta	
11	• Encryption • Decryption	7zAES	depend on pycryptodomex
12	• Unsupported	BCJ2	
13		Deflate64	

- A feature handling symbolic link is basically compatible with 'p7zip' implementation, but not work with original 7-zip because the original does not implement the feature.

2.4 Possible filters value

Here is a list of examples for possible filters values. You can use it when creating SevenZipFile object.

```
from py7zr import FILTER_LZMA, SevenZipFile

filters = [{'id': FILTER_LZMA}]
archive = SevenZipFile('target.7z', mode='w', filters=filters)
```

LZMA2 + Delta

```
[{'id': FILTER_DELTA}, {'id': FILTER_LZMA2, 'preset': PRESET_DEFAULT}]
```

LZMA2 + BCJ

```
[{'id': FILTER_X86}, {'id': FILTER_LZMA2, 'preset': PRESET_DEFAULT}]
```

LZMA2 + ARM

```
[{'id': FILTER_ARM}, {'id': FILTER_LZMA2, 'preset': PRESET_DEFAULT}]
```

LZMA + BCJ

```
[{'id': FILTER_X86}, {'id': FILTER_LZMA}]
```

LZMA2

```
[{'id': FILTER_LZMA2, 'preset': PRESET_DEFAULT}]
```

LZMA

```
[{'id': FILTER_LZMA}]
```

BZip2

```
[{'id': FILTER_BZIP2}]
```

Deflate

```
[{'id': FILTER_DEFLATE}]
```

ZStandard

```
[{'id': FILTER_ZSTD, 'level': 3}]
```

PPMd

```
[{'id': FILTER_PPMD, 'order': 6, 'mem': 24}]
```

```
[{'id': FILTER_PPMD, 'order': 6, 'mem': "16m"}]
```

Brolti

```
[{'id': FILTER_BROTLI, 'level': 11}]
```

7zAES + LZMA2 + Delta

```
[{'id': FILTER_DELTA}, {'id': FILTER_LZMA2, 'preset': PRESET_DEFAULT}, {'id':
FILTER_CRYPTO_AES256_SHA256}]
```

7zAES + LZMA2 + BCJ

```
[{'id': FILTER_X86}, {'id': FILTER_LZMA2, 'preset': PRESET_DEFAULT}, {'id':
FILTER_CRYPTO_AES256_SHA256}]
```

7zAES + LZMA

```
[{'id': FILTER_LZMA}, {'id': FILTER_CRYPTO_AES256_SHA256}]
```

7zAES + Deflate

```
[{'id': FILTER_DEFLATE}, {'id': FILTER_CRYPTO_AES256_SHA256}]
```

7zAES + BZip2

```
[{'id': FILTER_BZIP2}, {'id': FILTER_CRYPTO_AES256_SHA256}]
```

7zAES + ZStandard

```
[{'id': FILTER_ZSTD}, {'id': FILTER_CRYPTO_AES256_SHA256}]
```

CHAPTER

THREE

CONTRIBUTOR GUIDE

3.1 Development environment

If you're reading this, you're probably interested in contributing to py7zr. Thank you very much! The purpose of this guide is to get you to the point where you can make improvements to the py7zr and share them with the rest of the team.

3.1.1 Setup Python

The py7zr is written in the Python programming language. Python installation for various platforms with various ways. You need to install Python environment which support *pip* command. Venv/Virtualenv is recommended for development.

We have a test suite with python 3.6, 3.7, 3.8 and pypy3. If you want to run all the test with these versions and variant on your local, you should install these versions. You can run test with CI environment on Github actions.

3.1.2 Get Early Feedback

If you are contributing, do not feel the need to sit on your contribution until it is perfectly polished and complete. It helps everyone involved for you to seek feedback as early as you possibly can. Submitting an early, unfinished version of your contribution for feedback in no way prejudices your chances of getting that contribution accepted, and can save you from putting a lot of work into a contribution that is not suitable for the project.

3.2 Code Contributions

3.2.1 Steps submitting code

When contributing code, you'll want to follow this checklist:

1. Fork the repository on GitHub.
2. Run the tox tests to confirm they all pass on your system. If they don't, you'll need to investigate why they fail. If you're unable to diagnose this yourself, raise it as a bug report.
3. Write tests that demonstrate your bug or feature. Ensure that they fail.
4. Make your change.
5. Run the entire test suite again using tox, confirming that all tests pass including the ones you just added.
6. Send a GitHub Pull Request to the main repository's master branch. GitHub Pull Requests are the expected method of code collaboration on this project.

3.2.2 Code review

Contribution will not be merged until they have been code reviewed. There are limited reviewer in the team, reviews from other contributors are also welcome. You should implemented a review feedback unless you strongly object to it.

3.2.3 Code style

The py7zr uses the PEP8 code style. In addition to the standard PEP8, we have an extended guidelines

- line length should not exceed 125 charactors.
- It also use MyPy static type check enforcement.

3.3 Profiling

3.3.1 CPU and memory profiling

Run-time memory errors and leaks are among the most difficult errors to locate and the most important to correct. Memory profiling is used to detect memory leaks or unwanted memory usages.

It is also a difficult work to improve performance. CPU profiling help us to understand where is a hot spot of execution of a program.

3.3.2 mprofile

mprofile is a tool to do a memory profiling task for python. py7zr project has a test configuration for the memory profiling.

```
env PYTEST_ADDOPTS=--run-slow tox -e mprof
```

This example run all the test cases includes conditions which requires running duration.

After running test, you can find a chart in project root. *memory-profiile.png* and raw data as *mprofile_yyyyMMddhhmmss.dat*

3.4 Class and module design

The py7zr take class design that categorized into several sub modules to reflect its role.

The main class is py7zr.SevenZipFile() class which provide API for library users. The main internal classes are in the submodule py7zr.archiveinfo, which takes class structure as same as .7z file format structure.

Another important submodule is py7zr.compressor module that hold all related compression and encryption proxy classes for corresponding libraries to convert various interfaces into common ISevenZipCompressor() and ISevenZipDecompressor() interface.

All UI related classes and functions are separated from core modules. cli submodule is a place for command line functions and pretty printings.

Here is a whole classes diagram. There are part by part descriptions at Next sections.

3.4.1 Header classes

Header related classes are in py7zr.archiveinfo submodule.

3.4.2 Compressor classes

There are compression related classes in py7zr.compressor submodule.

3.4.3 IO Abstraction classes

There are two IO abstraction classes to provide Mem API and check method.

3.4.4 Callback classes

Here is a callback interface class. ExtractCallback class is a concrete class used in CLI.

3.5 Classes details

Here is a detailed interface documentation for implementer.

3.5.1 ArchiveFile Objects

Read 7zip format archives.

class py7zr.py7zr.**ArchiveFile**(*id: int*, *file_info: Dict[str, Any]*)

Represent each files metadata inside archive file. It holds file properties; filename, permissions, and type whether it is directory, link or normal file.

Instances of the *ArchiveFile* class are returned by iterating `files_list` of *SevenZipFile* objects. Each object stores information about a single member of the 7z archive. Most of users use `extractall()`.

The class also hold an archive parameter where file is exist in archive file folder(container).

property archivable: bool

File has a Windows *archive* flag.

property compressed: int | None

Compressed size

property crc32: int | None

CRC of archived file(optional)

property emptystream: bool

True if file is empty(0-byte file), otherwise False

file_properties() → Dict[str, Any]

Return file properties as a hash object. Following keys are included: 'readonly', 'is_directory', 'posix_mode', 'archivable', 'emptystream', 'filename', 'creationtime', 'lastaccesstime', 'lastwritetime', 'attributes'

property filename: str

return filename of archive file.

has_strdata() → bool

True if file content is set by writestr() method otherwise False.

property is_directory: bool

True if file is a directory, otherwise False.

property is_junction: bool

True if file is a junction/reparse point on windows, otherwise False.

property is_socket: bool

True if file is a socket, otherwise False.

property is_symlink: bool

True if file is a symbolic link, otherwise False.

property lastwritetime: *ArchiveTimestamp* | None

Return last written timestamp of a file.

property posix_mode: int | None

posix mode when a member has a unix extension property, or None :return: Return file stat mode can be set by os.chmod()

property readonly: bool

True if file is readonly, otherwise False.

property st_fmt: int | None

Returns

Return the portion of the file mode that describes the file type

class py7zr.py7zr.**ArchiveFileList**(*offset: int = 0*)

Iteratable container of ArchiveFile.

class py7zr.py7zr.**ArchiveInfo**(*filename: str*, *stat: stat_result*, *header_size: int*, *method_names: List[str]*, *solid: bool*, *blocks: int*, *uncompressed: List[int]*)

Hold archive information

class py7zr.py7zr.**FileInfo**(*filename*, *compressed*, *uncompressed*, *archivable*, *is_directory*, *creationtime*, *crc32*)

Hold archived file information.

class py7zr.py7zr.**SevenZipFile**(*file: BinaryIO | str | Path*, *mode: str = 'r'*, ***, *filters: List[Dict[str, int]] | None = None*, *dereference=False*, *password: str | None = None*, *header_encryption: bool = False*, *blocksize: int | None = None*, *mp: bool = False*)

The SevenZipFile Class provides an interface to 7z archives.

close()

Flush all the data into archive and close it. When close py7zr start reading target and writing actual archive file.

extractall(*path: Any | None = None*, *callback: ExtractCallback | None = None*) → None

Extract all members from the archive to the current working directory and set owner, modification time and permissions on directories afterwards. `path` specifies a different directory to extract to.

getnames() → List[str]

Return the members of the archive as a list of their names. It has the same order as the list returned by getmembers().

list() → List[*FileInfo*]

Returns contents information

reset() → None

When read mode, it reset file pointer, decompress worker and decompressor

write(*file: Path | str*, *arcname: str | None = None*)

Write single target file into archive.

writeall(*path: Path | str*, *arcname: str | None = None*)

Write files in target path into archive.

class py7zr.py7zr.**Worker**(*files*, *src_start: int*, *header*, *mp=False*)

Extract worker class to invoke handler.

archive(*fp: BinaryIO*, *files*, *folder*, *deref=False*)

Run archive task for specified 7zip folder.

decompress(*fp: BinaryIO*, *folder*, *fq: IO[Any]*, *size: int*, *compressed_size: int | None*, *src_end: int*) → int

decompressor wrapper called from extract method.

Parameters

- **fp** – archive source file pointer
- **folder** – Folder object that have decompressor object.
- **fq** – output file pathlib.Path
- **size** – uncompressed size of target file.
- **compressed_size** – compressed size of target file.
- **src_end** – end position of the folder

:returns None

extract(*fp: BinaryIO*, *path: Path | None*, *parallel: bool*, *skip_notarget=True*, *q=None*) → None

Extract worker method to handle 7zip folder and decompress each files.

extract_single(*fp: BinaryIO | str*, *files*, *path*, *src_start: int*, *src_end: int*, *q: Queue | None*, *exc_q: Queue | None = None*, *skip_notarget=True*) → None

Single thread extractor that takes file lists in single 7zip folder.

register_filelike(*id: int*, *fileish: MemIO | Path | None*) → None

register file-ish to worker.

py7zr.py7zr.**is_7zfile**(*file: BinaryIO | str | Path*) → bool

Quickly see if a file is a 7Z file by checking the magic number. The file argument may be a filename or file-like object too.

py7zr.py7zr.**pack_7zarchive**(*base_name*, *base_dir*, *owner=None*, *group=None*, *dry_run=None*, *logger=None*)

Function for registering with shutil.register_archive_format().

py7zr.py7zr.**unpack_7zarchive**(*archive*, *path*, *extra=None*)

Function for registering with shutil.register_unpack_format().

3.5.2 archiveinfo module

class py7zr.archiveinfo.**Bond**(*incoder*, *outcoder*)

Represent bindings between two methods. bonds[i] = (incoder, outstream) means methods[i].stream[outstream] output data go to method[incoder].stream[0]

class py7zr.archiveinfo.**FilesInfo**

holds file properties

class py7zr.archiveinfo.**Folder**

a "Folder" represents a stream of compressed data. coders: list of coder num_coders: length of coders coder: hash list keys of coders: method, numinstreams, numoutstreams, properties unpacksizes: uncompressed sizes of outstreams

class py7zr.archiveinfo.**Header**

the archive header

class py7zr.archiveinfo.**HeaderStreamsInfo**

Header version of StreamsInfo

class py7zr.archiveinfo.**PackInfo**

information about packed streams

class py7zr.archiveinfo.**SignatureHeader**

The SignatureHeader class hold information of a signature header of archive.

class py7zr.archiveinfo.**StreamsInfo**

information about compressed streams

class py7zr.archiveinfo.**SubstreamsInfo**

defines the substreams of a folder

class py7zr.archiveinfo.**UnpackInfo**

combines multiple folders

class py7zr.archiveinfo.**WriteWithCrc**(*fp: BinaryIO*)

Thin wrapper for file object to calculate crc32 when write called.

tell()

Return current stream position.

py7zr.archiveinfo.**read_real_uint64**(*file: BinaryIO*) → Tuple[int, bytes]

read 8 bytes, return unpacked value as a little endian unsigned long long, and raw data.

py7zr.archiveinfo.**read_uint32**(*file: BinaryIO*) → Tuple[int, bytes]

read 4 bytes, return unpacked value as a little endian unsigned long, and raw data.

py7zr.archiveinfo.**read_uint64**(*file: BinaryIO*) → int

read UINT64, definition show in write_uint64()

py7zr.archiveinfo.**read_utf16**(*file: BinaryIO*) → str

read a utf-16 string from file

py7zr.archiveinfo.**write_real_uint64**(*file: BinaryIO | WriteWithCrc*, *value: int*)

write 8 bytes, as an unsigned long long.

py7zr.archiveinfo.**write_uint32**(*file: BinaryIO | WriteWithCrc*, *value*)

write uint32 value in 4 bytes.

py7zr.archiveinfo.**write_uint64**(*file: BinaryIO | WriteWithCrc*, *value: int*)

UINT64 means real UINT64 encoded with the following scheme:

Size of encoding sequence depends from first byte:
First_Byte Extra_Bytes Value
(binary)
0xxxxxxx : (xxxxxxx)
10xxxxxx BYTE y[1] : (xxxxxx << (8 * 1)) + y
110xxxxx BYTE y[2] : (xxxxx << (8 * 2)) + y
...
1111110x BYTE y[6] : (x << (8 * 6)) + y
11111110 BYTE y[7] : y
11111111 BYTE y[8] : y

py7zr.archiveinfo.**write_utf16**(*file: BinaryIO | WriteWithCrc*, *val: str*)

write a utf-16 string to file

3.5.3 compressor module

class py7zr.compressor.**AESCompressor**(*password: str*, *blocksize: int | None = None*)

AES Compression(Encryption) class. It accept pre-processing filter which may be a LZMA compression.

compress(*data*)

Compression + AES encryption with 16byte alignment.

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**AESDecompressor**(*aes_properties: bytes*, *password: str*, *blocksize: int | None = None*)

Decrypt data

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**BCJDecoder**(*size: int*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**BCJEncoder**

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**BcjArmDecoder**(*size: int*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**BcjArmEncoder**

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**BcjArmtDecoder**(*size: int*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**BcjArmtEncoder**

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**BcjPpcDecoder**(*size: int*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**BcjPpcEncoder**

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**BcjSparcDecoder**(*size: int*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**BcjSparcEncoder**

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**BrotliCompressor**(*level*)

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush() → bytes

Flush output buffer(interface) :return: output data

class py7zr.compressor.**BrotliDecompressor**(*properties: bytes*, *block_size: int*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*)

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**CopyCompressor**

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**CopyDecompressor**

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**Deflate64Compressor**

compress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Compress data (interface) :param data: input data :return: output data

flush() → bytes

Flush output buffer(interface) :return: output data

class py7zr.compressor.**Deflate64Decompressor**

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**DeflateCompressor**

compress(*data*)

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**DeflateDecompressor**

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**ISevenZipCompressor**

abstract compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

abstract flush() → bytes

Flush output buffer(interface) :return: output data

class py7zr.compressor.**ISevenZipDecompressor**

abstract decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**LZMA1Compressor**(*filters*)

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush() → bytes

Flush output buffer(interface) :return: output data

class py7zr.compressor.**LZMA1Decompressor**(*filters*, *unpacksize*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**MethodsType**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

class py7zr.compressor.**PpmdCompressor**(*properties: bytes*)

Compress with PPMd compression algorithm

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush()

Flush output buffer(interface) :return: output data

class py7zr.compressor.**PpmdDecompressor**(*properties: bytes*, *blocksize: int | None = None*)

Decompress PPMd compressed data

decompress(*data: bytes | bytearray | memoryview*, *max_length=-1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

class py7zr.compressor.**SevenZipCompressor**(*filters=None*, *password=None*, *blocksize: int | None = None*)

Main compressor object to configured for each 7zip folder.

class py7zr.compressor.**SevenZipDecompressor**(*coders: List[Dict[str, Any]]*, *packsize: int*, *unpacksizes: List[int]*, *crc: int | None*, *password: str | None = None*, *blocksize: int | None = None*)

Main decompressor object which is properly configured and bind to each 7zip folder. because 7zip folder can have a custom compression method

class py7zr.compressor.**SupportedMethods**

Hold list of methods.

class py7zr.compressor.**ZstdCompressor**(*level: int*)

compress(*data: bytes | bytearray | memoryview*) → bytes

Compress data (interface) :param data: input data :return: output data

flush() → bytes

Flush output buffer(interface) :return: output data

class py7zr.compressor.**ZstdDecompressor**(*properties: bytes*, *blocksize: int*)

decompress(*data: bytes | bytearray | memoryview*, *max_length: int = -1*) → bytes

Decompress data (interface) :param data: input data :param max_length: maximum length of output data when it can respect, otherwise ignore. :return: output data

3.5.4 helpers module

class py7zr.helpers.**ArchiveTimestamp**

Windows FILETIME timestamp.

as_datetime()

Convert FILETIME to Python datetime object.

totimestamp() → float

Convert 7z FILETIME to Python timestamp.

exception py7zr.helpers.**BufferOverflow**

class py7zr.helpers.**LocalTimezone**

dst(*dt*)

datetime -> DST offset as timedelta positive east of UTC.

fromutc(*dt*)

datetime in UTC -> datetime in local time.

tzname(*dt*)

datetime -> string name of time zone.

utcoffset(*dt*)

datetime -> timedelta showing offset from UTC, negative values indicating West of UTC

class py7zr.helpers.**MemIO**(*buf: BinaryIO*)

pathlib.Path-like IO class to write memory(io.Bytes)

class py7zr.helpers.**NullIO**

pathlib.Path-like IO class of /dev/null

class py7zr.helpers.**UTC**

dst(*dt*)

datetime -> DST offset as timedelta positive east of UTC.

tzname(*dt*)

datetime -> string name of time zone.

utcoffset(*dt*)

datetime -> timedelta showing offset from UTC, negative values indicating West of UTC

py7zr.helpers.**calculate_crc32**(*data: bytes*, *value: int = 0*, *blocksize: int = 1048576*) → int

Calculate CRC32 of strings with arbitrary lengths.

py7zr.helpers.**calculate_key**(*password: bytes*, *cycles: int*, *salt: bytes*, *digest: str*) → bytes

Calculate 7zip AES encryption key. Concat values in order to reduce number of calls of Hash.update().

py7zr.helpers.**canonical_path**(*target: PurePath*) → PurePath

Return a canonical path of target argument.

py7zr.helpers.**check_archive_path**(*arcname: str*) → bool

Check arcname argument is valid for archive. It should not be absolute, if so it returns False. It should not be evil traversal attack path. Otherwise, returns True.

py7zr.helpers.**filetime_to_dt**(*ft*)

Convert Windows NTFS file time into python datetime object.

py7zr.helpers.**get_sanitized_output_path**(*fname: str*, *path: Path | None*) → Path

check f.filename has invalid directory traversals When condition is not satisfied, raise Bad7zFile

py7zr.helpers.**is_path_valid**(*target: Path*, *parent: Path*) → bool

Check if target path is valid against parent path. It returns False when target path has '..' and point out of parent path. Otherwise, returns True.

py7zr.helpers.**is_relative_to**(*my: PurePath*, **other*) → bool

Return True when path is relative to other path, otherwise False.

py7zr.helpers.**islink**(*path*)

Cross-platform islink implementation. Support Windows NT symbolic links and reparse points.

py7zr.helpers.**readlink**(*path: str | Path*, *, *dir_fd=None*) → str | Path

Cross-platform compat implementation of os.readlink and Path.readlink(). Support Windows NT symbolic links and reparse points. When called with path argument as pathlike(str), return result as a pathlike(str). When called with Path object, return also Path object. When called with path argument as bytes, return result as a bytes.

py7zr.helpers.**remove_relative_path_marker**(*path: str*) → str

Removes './' from the beginning of a path-like string

CHAPTER

FOUR

.7Z FORMAT SPECIFICATION

4.1 Abstract

7-zip archive is one of popular files compression and archive formats. There has been no well-defined file format because there is no precise specification document in 20 years from its birth, so it has been considered as an application proprietary format.

There are some independent implementation of utility to handle 7-zip archives, precise documentation of specification is mandatory to keep compatibility and interoperability among implementations.

This specification defines an archive file format of .7z archive. A purpose of this document is to provide a concrete documentation to archive an interoperability among implementations.

4.2 Copyright Notice

4.3 Introduction

4.3.1 Purpose

This specification is intended to define a cross-platform, interoperable file storage and transfer format. The information here is meant to be a concise guide for those wishing to implement libraries and utility to handle 7-zip archive files.

This documentations is NOT a specification of any existed utilities and libraries. This documentation does not have some features which is implemented in an existed utility. It is because this document purpose is to keep interoperability.

4.3.2 Intended audience

This specification is intended for use by implementors of software to compress files into 7-zip format and/or decompress files from 7-zip format.

The text of the specification assumes a basic background in programming at the level of bits and other primitive data representations.

4.3.3 Scope

"7-zip archive" is one of popular files compression and archive formats. It is universally used to aggregate, compress, and encrypt files into a single interoperable container. No specific use or application need is defined by this format and no specific implementation guidance is provided. This document provides details on the storage format for creating 7-zip files. Information is provided on the records and fields that describe what a .7z file is.

This specification does not provide technical specification of compression methods such as LZMA, LZMA2, Delta, BCJ and every other methods. It also does not provide technical specification of encryption and hash methods such as AES and SHA256.

4.3.4 Trademarks

7-zip is a public-domain utility on Microsoft Windows platforms written by Igor Pavlov. 7-zip archive file format was originally produced and defined by 7-zip utility. p7zip is a cross-platform utility to handle 7zip archive file, which is a port of 7-zip to posix. py7zr is a library and utility written with pure python3 to handle 7zip archive, that is distributed under GNU Lessaer General Public License version 2.1 and later. xzutils is an file compression/decompression utility. liblzma is a library to provide LZMA and LZMA2 compression algorithm provided by xzutils project. Python is one of popular computer language and running platform copyrighted and licensed by Python Foundation. Python 3 provide lzma API deppend on liblzma.

4.4 Motivation

There are several file archive format and utilities. Many of them are born as proprietary format of archive utility software, because of its nature, only standardized formats are now alived as portable, stable for long time and freely usable specification. PKWare ZIP, GNU Tar and GZip are examples for it. Since 7-zip, its format and algorithm LZMA/LZMA2 are born as public-domain in 1999, it has been known as one of long lived file format.

There are two effort to make .7z archives as well-documented, portable, and long life. One is a documentation project here, and other is a software development project to be compatible with original 7zip and p7zip utility such as py7zr.

4.5 Notations

- Use of the term SHALL indicates a required element.
- MAY NOT or SHALL NOT indicates an element is prohibited from use.
- SHOULD indicates a RECOMMENDED element.
- SHOULD NOT indicates an element NOT RECOMMENDED for use.
- MAY indicates an OPTIONAL element.

4.6 Data Representations

This chapter describes basic data representations used in 7-zip file.

4.6.1 BYTE

BYTE is a basic data type to store a char, Property ID or bitfield.

4.6.2 BYTEARRAY

BYTEARRAY is a sequence of BYTE. Its length SHALL be defined in another place.

4.6.3 String

There are two type of string data is used in 7-zip archive format.

- UTF-16-LE
- UTF-8

4.6.4 Integers

All integers that require more than one byte SHALL be in a little endian, Least significant byte (LSB) comes first, then more significant bytes in ascending order of significance (LSB MSB for two byte integers, B0 B1 B2 B3 for four bytes integers). The highest bit (value 128) of byte is number bit 7 and lowest bit (value 1) is number bit 0. Values are unsigned unless otherwise noted.

name	size	description
UINT32	4 bytes	integer at little endian represent 0 to 4,294,967,295 (`0xffffffff`)
UINT64	8 bytes	integer at little endian represent 0 to 18,446,744,073,709,551,615 (`0xffffffffffffffff`)
NUMBER	1-9 bytes	variable length integer value represent 0 to 18,446,744,073,709,551,615 (`0xffffffffffffffff`)

NUMBER SHALL be a integer value encoded with the following scheme. in byte length between one byte to nine bytes.

Size of encoding sequence SHALL indicated at first byte. The rest bits of first byte represent a bits from MSB of value. Following bytes SHOULD be an integer as little endian.

First_Byte (binary)	Extra_Bytes	Value (y: little endian integer)
0xxxxxxx		(0b0xxxxxxx)
10xxxxxx	BYTE y[1]	(0b00xxxxxx << (8 * 1)) + y
110xxxxx	BYTE y[2]	(0b000xxxxx << (8 * 2)) + y
1110xxxx	BYTE y[3]	(0b0000xxxx << (8 * 3)) + y
11110xxx	BYTE y[4]	(0b00000xxx << (8 * 4)) + y
111110xx	BYTE y[5]	(0b000000xx << (8 * 5)) + y
1111110x	BYTE y[6]	(0b0000000x << (8 * 6)) + y
11111110	BYTE y[7]	y
11111111	BYTE y[8]	y

4.6.5 BitField

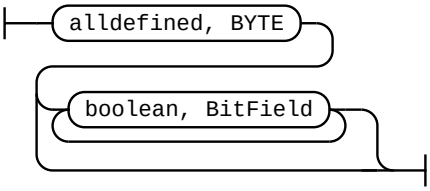
BitField represent eight boolean values in single BYTE.

The bit field is defined which order is from MSB to LSB, i.e. bit 7 (MSB) of first byte indicate a boolean for first stream, object or file, bit 6 of first byte indicate a boolean for second stream, object or file, and bit 0(LSB) of second byte indicate a boolean for 16th stream, object or file.

A length is vary according to a number of items to indicate. If a number of items is not multiple of eight, rest of bitfield SHOULD zero.

4.6.6 BooleanList

BooleanList is a list of boolean bit arrays. It has two field. First it defines an existence of boolean values for each items of number of files or objects. Then boolean bit fields continues. There is an extension of expression that indicate all boolean values is True, and skip boolean bit fields.



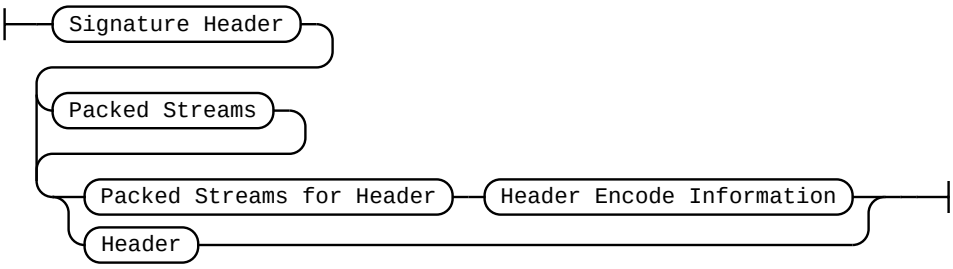
4.7 File format

7-zip archive file format SHALL consist of three part. 7-zip archive file SHALL start with signature header. The data block SHOULD placed after the signature header. The data block is shown as Packed Streams.

A header database SHOULD be placed after the data block. The data block MAY be empty when no archived contents exists. So Packed Streams is optional.

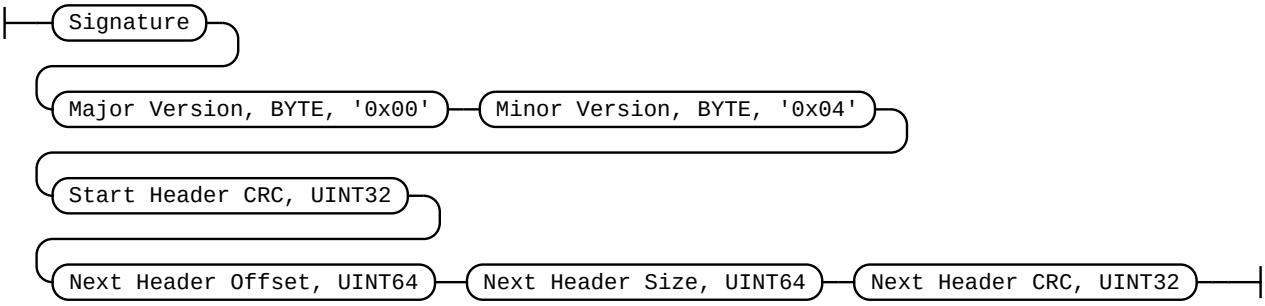
Since Header database CAN be encoded then it SHOULD place after data block, that is Packed Streams for Headers. When Header database is encoded, Header encode Information SHALL placed instead of Header.

When Header database is placed as plain form, Packed Streams for Headers SHALL NOT exist.



4.7.1 Signature Header

Signature header SHALL consist in 32 bytes. Signature header SHALL start with Signature then continues with archive version. Start Header SHALL follow after archive version.



It can be observed as follows when taken hex dump.

address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Signature						VN		S.H. CRC				N.H. offset			
0x0010	offset(cont)				N.H. size								N.H. CRC			

Signature

The first six bytes of a 7-zip file SHALL always contain `b'7z\\xbc\\xaf\\x27\\x1c'`.

Version Number

Version number SHALL consist with two bytes. Major version is `0x00`, and minor version is `0x04` for now.

Start Header CRC

It SHALL be stored in form of `UINT32`. This CRC value SHALL be calculated from Next Header Offset, Next Header size and Next Header CRC.

Next Header offset

Next header offset SHALL be an offset from end of signature header to header database. Because signature header always consist with 32 bytes, the offset SHOULD be a value that absolute position of header database in archive file - 32 bytes. Next header offset SHALL be stored as `UINT64`.

Next Header size

Next header size SHALL be an size of a header database. Because a header database MAY be encoded, Next header size SHALL consist of encoded(packed) size, not a raw size. Next header size SHALL be stored as `UINT64`.

Next Header CRC

Next header CRC SHALL a `CRC32` of Header that SHALL be stored in `UINT32`.

4.7.2 Property IDs

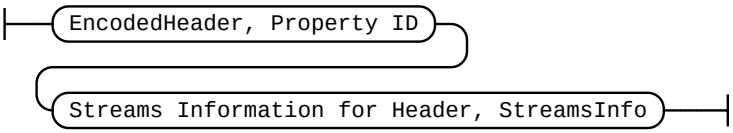
Information stored in Header SHALL be placed after Property ID. For example, Header Info block start with 0x01, which means Header, then continues data blocks, and `0x00`, which is `END`, is placed at last. This structure can be recursive but there is a rules where particular ID can exist.

Property ID SHALL be a BYTE.

ID	Property
0x00	END
0x01	Header
0x02	ArchiveProperties
0x03	AdditionalStreamsInfo
0x04	MainStreamsInfo
0x05	FilesInfo
0x06	PackInfo
0x07	UnPackInfo
0x08	SubStreamsInfo
0x09	Size
0x0A	CRC
0x0B	Folder
0x0C	CodersUnPackSize
0x0D	NumUnPackStream
0x0E	EmptyStream
0x0F	EmptyFile
0x10	Anti
0x11	Name
0x12	CTime
0x13	ATime
0x14	MTime
0x15	Attributes
0x16	Comment
0x17	EncodedHeader
0x18	StartPos
0x19	Dummy

4.7.3 Header encode Information

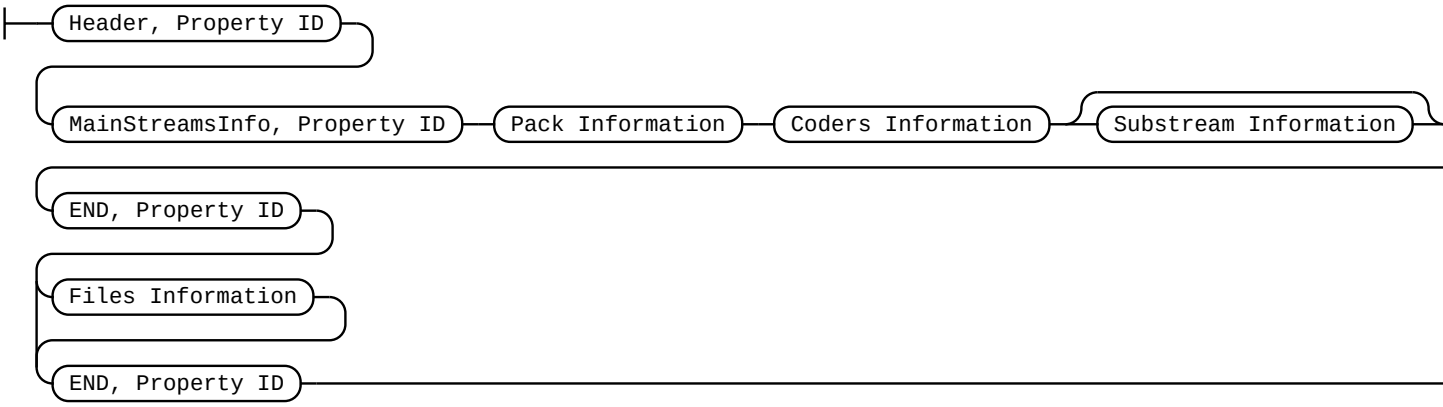
Header encode Information is a Streams Information data for Header data as encoded data followed after ID `0x17`, EncodedHeader Property.



4.7.4 Header

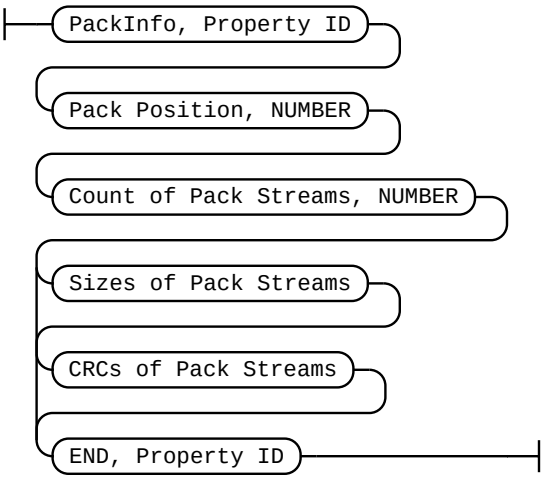
Header SHALL be consist of Main Streams. It MAY be also consist of file list information. It SHALL placed at a position where Start header offset pointed in archive file. Header database MAY be encoded.

When raw header is located, it SHOULD become the following structure. Raw header SHALL start with one byte ID `0x01`.



4.7.5 Pack Information

Pack Information SHALL start with one byte of id value; 0x06. Pack Information SHALL be const with Pack Position, Number of Pack Streams, a list of sizes of Pack Streams and a list of CRCs of pack streams. Pack position and Number of Pack streams SHALL be stored as variable length `NUMBER` form. Sizes of packed Streams SHALL stored as a list of `NUMBER`.



Pack Position

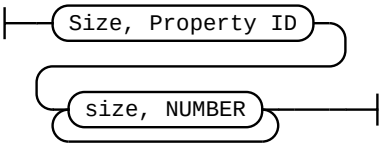
Pack Position SHALL indicate a position of encoded streams that value SHALL be an offset from the end of signature header. It MAY be a next position of end of signature header.

Count of Pack Streams

Count of Pack Streams SHALL indicate a number of encoded streams. LZMA and LZMA2 SHOULD have a single (one) stream. 7-zip CAN have encoding methods which produce multiple encoded streams. When there are multiple streams, a value of Number of Pack Streams SHALL indicate it.

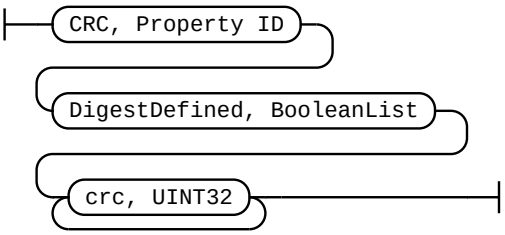
Sizes of Pack Streams

Sizes of Pack Streams SHOULD be omitted when Number of Pack Streams is zero. This is an array of `NUMBER` values which length is as same as Count of Pack Streams. Size SHALL be positive integer and SHALL stored in `NUMBER`.



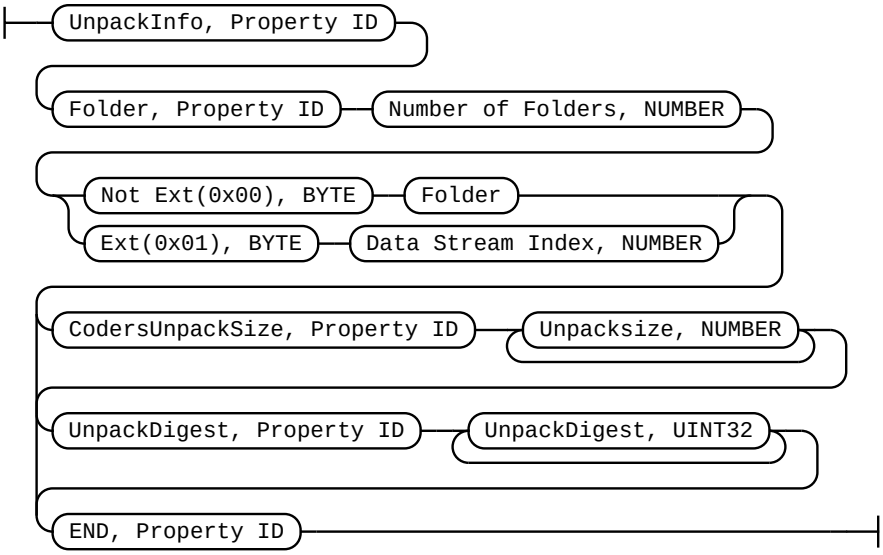
CRCs of Pack Streams

When Count of Pack Streams is zero, then CRCs of Pack Streams SHALL not exist. CRC CAN be exist and indicated as DigestDefined BooleanList. CRC SHALL be `CRC32` and stored in `UINT32`.



4.7.6 Coders Information

Coders Information SHALL located after Main Streams Information. It SHALL provide encoding and encryption filter parameters. It MAY be a single coder or multiple coders defined. It SHALL NOT be more than five coders. (Maximum four)



In default Folders information is placed inline, then External flag is `0x00`.

UnpackSizes

UnpackSizes is a list of decompress sizes for each archived file data. When extract data from the archive, it SHALL be distilled from unpack streams and split chunk into defined sizes.

Filenames are defined in File Information block. An order of data chunks and a order of filenames SHALL be same, except for filenames which is defined as empty stream.

UnpackDigests

UnpackDigests is a list of `CRC32` of decompress data digests for each folders. When extract data from the archive, it CAN check an integrity of data.

It SHALL be a list of `NUMBER` and its length SHALL be as same as number of folders. It MAY be skipped when Substreams Information defined.

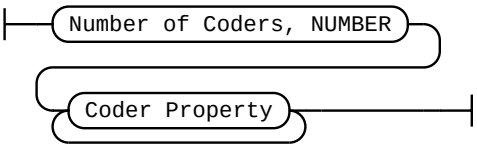
4.7.7 Folders

Folder in 7-zip archive means a basic container unit for encoded data. It brings encoded data. The data chunk Packed Streams is defined as series of Folders.

Each Folder has coder information. CoderInfo is consist of flag, number of streams and properties.

Flag indicate the coder is simple i.e. single input and single output, or complex i.e. multiple input, multiple output.

When simple coder, number of streams is always one for input, and one for output, so it SHALL be skipped.



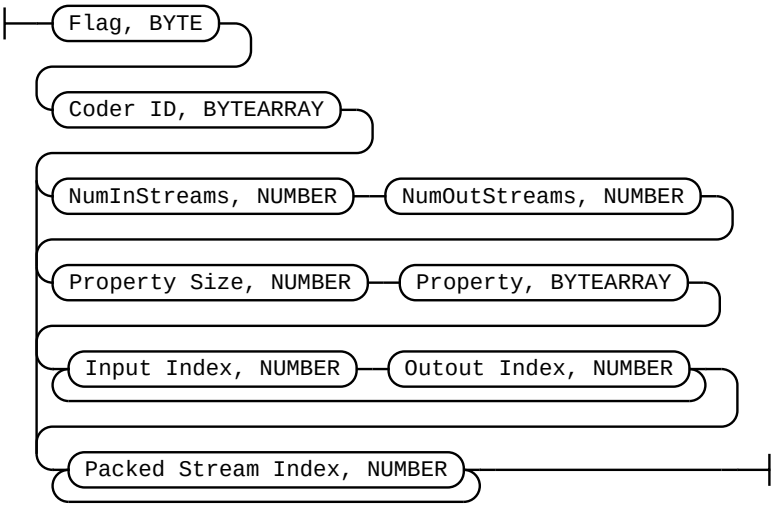
Number of coder SHALL be a NUMBER integer number. Coder Properties SHALL be a list of Coder Property with length SHALL be as same as Number of coder.

Coder Property

Coder Property is defined with flag which indicate coder types. According to flag that indicate coder is complex, the Coder Property MAY have a number of input and output streams of coder.

Flag is defined in one byte as following bit definitions.

- bit 3-0: Codec ID size
- bit 4: Is complex codec
- bit 5: There are attributes
- bit 6-7: Reserved, it SHOULD always be zero.



BindPairs

BindPairs describe connection among coders when coder produce multiple output or required multiple input.

A coder property format is vary with flag. Following pseudo code indicate how each parameter located for informative purpose.

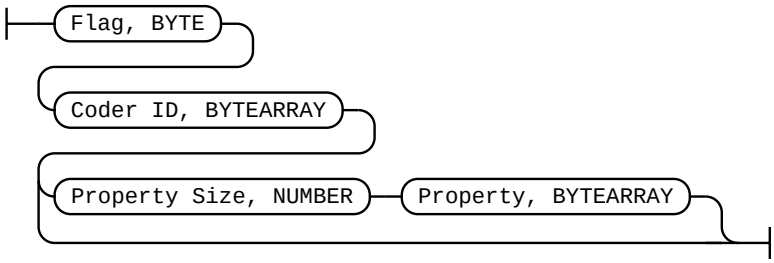
```
if (Is Complex Coder)
 {
   NUMBER ``NumInStreams``;
   NUMBER ``NumOutStreams``;
 }
 if (There Are Attributes)
 {
   NUMBER ``PropertiesSize``
```

(continues on next page)

(continued from previous page)

```
   BYTE ``Properties[PropertiesSize]``
 }
}
NumBindPairs :  = ``NumOutStreamsTotal`` - 1;
for (``NumBindPairs``)
 {
   NUMBER ``InIndex``;
   NUMBER ``OutIndex``;
 }
NumPackedStreams : ``NumInStreamsTotal`` - ``NumBindPairs``;
 if (``NumPackedStreams`` > 1)
   for(``NumPackedStreams``)
   {
     NUMBER ``Index``;
   };
```

When using only simple codecs, which has one input stream and one output stream, coder property become as simple as follows;



Here is an example of bytes of coder property when specifying LZMA.

- `b'\x23\x03\x01\x01\x05\x5D\x00\x10\x00\x00'`

In this example, first byte 0x23 indicate that coder id size is three bytes, and it is not complex codec and there is a codec property. A coder ID is `b'\x03\x01\x01'` and property length is five and property is `b'\x5D\x00\x10\x00\x00'`.

4.7.8 Codec IDs

Conformant implementations SHALL support mandatory codecs that are COPY, LZMA, LZMA2, BCJ, and Delta. There are a variant of BCJ that are X86, PowerPC, SPARC, ARM, ARMTHUMB, and IA64. Conformant implementations SHOULD also support optional codecs that are AES, BZIP2, DEFLATE, BCJ2, and PPMd. Implementations MAY support additional codecs that are ZStandard, and LZ4. It MAY also support proprietary codec such as DEFLATE64.

Conformant implementations SHALL accept these codec IDs and when it does not support it, it SHOULD report it as not supported.

Here is a list of famous codec IDs.

NAME	ID	Note
COPY	0x00	
DELTA	0x03	
BCJ	0x04	
LZMA	0x030101	
P7Z_BCJ	0x03030103	
P7Z_BCJ2	0x0303011b	[1] [2]
BCJ_PPC	0x03030205	
BCJ_IA64	0x03030301	
BCJ_ARM	0x03030501	
BCJ_ARMT	0x03030701	
BCJ_SPARC	0x03030805	
LZMA2	0x21	
BZIP2	0x040202	
DEFLATE	0x040108	
DEFLATE64	0x040109	Page 34, [3]
ZSTD	0x04f71101	
BROTLI	0x04f71102	
LZ4	0x04f71104	[1]
LZS	0x04f71105	[1]
LIZARD	0x04f71106	[1]
AES	0x06f10701	

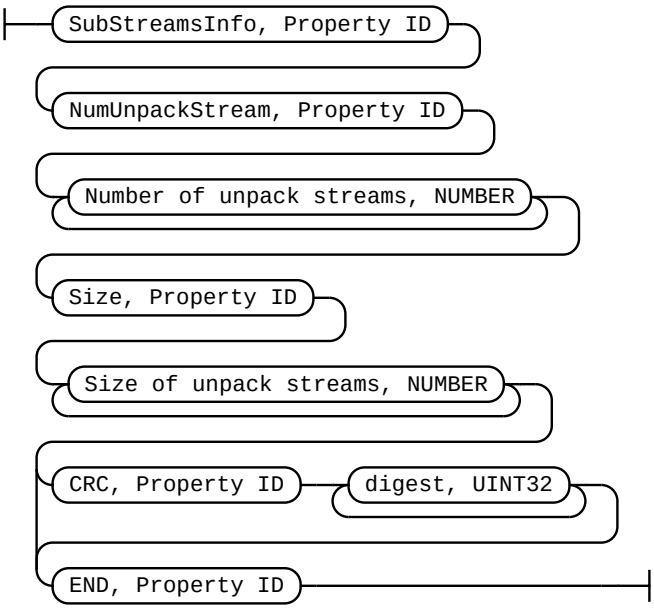
4.7.9 Substreams Information

Substream Information is an optional field that indicate substreams from each folder produces.

When the archive is not solid, there SHALL NOT be SubStreams information. When SubStreams Information is omitted, extractor still know a unpack size information as folder information.

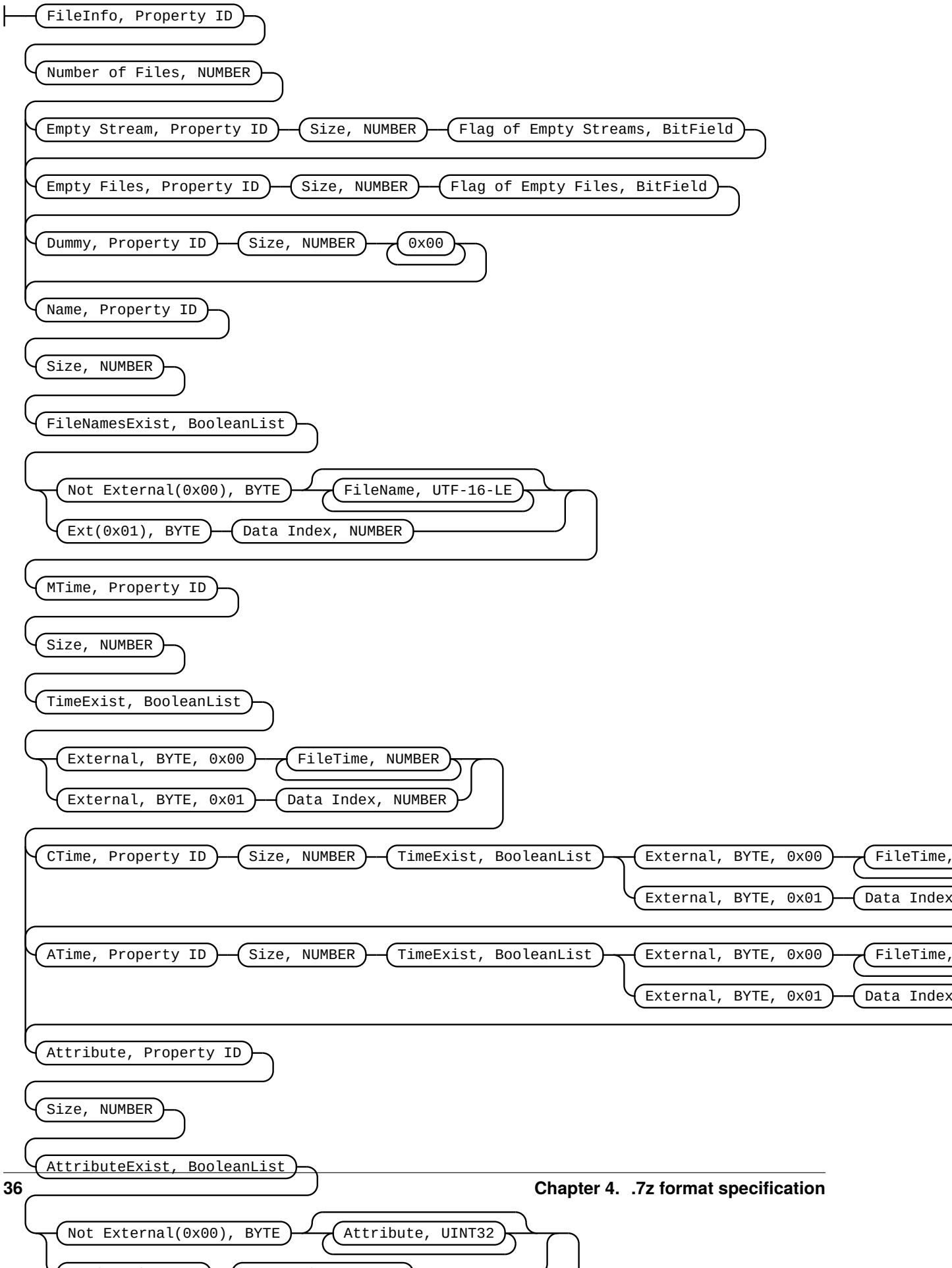
Substreams Information hold an information about archived data blocks as in extracted form. It SHALL exist that number of unpack streams, size of each unpack streams, and CRC of each streams

[1] Py7zr does not support BCJ2, DEFLATE64, LZ4, LZS and LIZARD
[2] There is no plan to support BCJ2 by py7zr since Python standard lzma module does not support.
[3] DEFLATE64 is supported only for extraction.



4.7.10 Files Information

Files Information SHOULD hold a list of files, directories and symbolic links. Its order SHALL be as same as order of streams defined in packed information. A type of file is stored in Attribute field.


FileInfo, Property ID
Number of Files, NUMBER
Empty Stream, Property ID
Size, NUMBER
Flag of Empty Streams, BitField
Empty Files, Property ID
Size, NUMBER
Flag of Empty Files, BitField
Dummy, Property ID
Size, NUMBER
0x00
Name, Property ID
Size, NUMBER
FileNamesExist, BooleanList
Not External(0x00), BYTE
FileName, UTF-16-LE
Ext(0x01), BYTE
Data Index, NUMBER
MTime, Property ID
Size, NUMBER
TimeExist, BooleanList
External, BYTE, 0x00
FileTime, NUMBER
External, BYTE, 0x01
Data Index, NUMBER
CTime, Property ID
Size, NUMBER
TimeExist, BooleanList
External, BYTE, 0x00
FileTime,
External, BYTE, 0x01
Data Index
ATime, Property ID
Size, NUMBER
TimeExist, BooleanList
External, BYTE, 0x00
FileTime,
External, BYTE, 0x01
Data Index
Attribute, Property ID
Size, NUMBER
AttributeExist, BooleanList
Not External(0x00), BYTE
Attribute, UINT32

Size

Size field indicate a size of next data. For example, Name size means, a size in byte from a start of FileNamesExist field and an end of file names.

Empty Streams

Empty streams has a number of emptystreams and a boolean list to indicate which file entry does not have a packed stream.

Dummy

Dummy MAY be placed for alignment. When processing File Names, which is UTF-16-LE, it is better to be aligned in word barrier.

FileName

FileNam SHALL be a wide character string encoded with UTF-16-LE and follows `wchar_t NULL` character, i.e. `0x0000.`

Path separator SHALL be normalized as '/', which is as POSIX standard. FileName SHOULD be relative path notation.

Attribute

Attribute is a `UINT32` integer value. From bit 0 to 15 are as same as Windows attributes. Bit 16 to 31 is used for storing unix attributes. When file is a symbolic link, it SHOULD has an attribute that `UNIX_EXTENSION` flag enabled, and link bit of unix attributes.

Table 1: Attribute values

ID/Value	Description
FILE_ATTRIBUT 1 (0x1)	A file that is read-only.
FILE_ATTRIBUT 2 (0x2)	The file or directory is hidden.
FILE_ATTRIBUT 16 (0x10)	It identifies a directory.
FILE_ATTRIBUT 32 (0x20)	A file or directory that is an archive file or directory.
FILE_ATTRIBUT 1024 (0x400)	file or directory that has an associated reparse point, or a file that is a symbolic link.
bit 16-31	UNIX file permissions and attributes. 16bit shift to left of permissions and attributes.
UNIX_EXTENSI (0x8000)	Indicate a unix permissions and file attributes are bundled when 1.

FileTime

FileTime are `NUMBER` values in 100-nanosecond intervals since 1601/01/01 (UTC)

4.8 File type and a way

4.8.1 Normal files

Normal files are stored with packed streams and ordinal file information. Its contents are stored into packed stream. It SHOULD have an attribute of Windows such as `FILE_ATTRIBUTE_ARCHIVE`. It MAY also have an attribute of UNIX such as `rw_r__r__` permissions.

4.8.2 Empty files

Empty files, which size is zero, SHALL be stored without packed stream, and with flle information. It SHOULD have an attribute of Windows such as `FILE_ATTRIBUTE_ARCHIVE`. It MAY also have an attribute of UNIX such as `rw_r__r__` permissions.

4.8.3 Directories

Directories are stored without packed streams. It have entries in file information. It SHALL have an attribute which is `FILE_ATTRIBUTE_DIRECTORY`. It MAY also have an attribute of UNIX such as `rwxr_xr_x` permissions.

4.8.4 Special Files

There is an extension to handle special files such as sockets, device files, and symbolic links. A type of special files is indicated as file attribute. Further attribute of special file is stored as a content.

Compliant client MAY skip record of special files on extraction.

Symbolic links

Symbolic links are stored as packed streams and file information. Its target file path, in relative, are recorded into packed streams in UTF-8 character encoding. It SHALL have a UNIX attribute which is `S_IFLNK`.

REPARSE_POINT on Windows

Reparse point on windows SHOULD be stored with packed stream and file information. Its target link path, in absolute, are recorded into packed stream in UTF-8 character encoding. It SHALL have an attribute which is `FILE_ATTRIBUTE_REPARSE_POINT`.

4.9 Appendix: BNF expression (Informative)

This clause shows extended BNF expression of 7-zip file format.

```
7-zip archive        ::=   SignatureHeader, [PackedStreams],
                           [PackedStreamsForHeaders], Header | HeaderInfo
SignatureHeader      ::=   Signature, ArchiveVersion, StartHeader
Signature            ::=   ``b'7z\xBC\xAF\x27\x1C'``
ArchiveVersion       ::=   ``b'\x00\x04'``
StartHeader          ::=   StartHeaderCRC, NextHeaderOffset,
                           NextHeaderSize, NextHeaderCRC
NextHeaderOffset     ::=   `UINT64`
NextHeaderSize       ::=   `UINT64`
NextHeaderCRC        ::=   `UINT32`
StreamsInfo          ::=   PackInfo, CodersInfo, SubStreamsInfo
PackInfo             ::=   `0x06`, PackPos, NumPackStreams,
                           SizesOfPackStream, CRCsOfPackStreams
CodersInfo           ::=   `0x07`, FoldersInfo
Folders Information  ::=   0x0B, NumFolders, FolderInfo,
                           CoderUnpackSizes, UnpackDigests, 0x00
FoldersInfo          ::=   `0x0B`, NumFolders, (`0x00`, Folders) | (`0x01`, DataStreamIndex)
                           [`0x0C`, UnPackSizes, [`0x0A`, UnpackDigests]], `0x00`
Folders              ::=   Folder{ Number of Folders }
UnpackSizes          ::=   UnPackSize { Sum of NumOutStreams for each Folders }
UnpackSize           ::=   `NUMBER`
UnpackDigests        ::=   CRC32 { Number of folders }
SubStreamsInfo       ::=   `0x08`, `0x0D`, NumUnPackStreamsInFolders{Num of Folders],
                           `0x09`, UnPackSize, `0x0A`,
                           Digests{Number of streams with unknown CRC}, 0x00
Folder               ::=   NumCoders, CoderData { NumCoders }
CoderData            ::=   CoderFlag, CoderID, NumCoderStreamInOut, Properties,
                           BinPairs, PackedStreamIndex
CoderFlag            ::=   BYTE(bit 0:3 CodecIdSize, 4:  Is Complex Coder,
                           5:  There Are Attributes, 6:  Reserved, 7:  0)
CoderId              ::=   BYTE{CodecIdSize}
FilesInfo            ::=   `0x05`, NumFiles, FileInfo, [FileInfo]
FileInfo             ::=   NumFiles, [0x0E, bit array of IsEmptyStream],
                           [`0x0F`, bit array of IsEmptyFile],
                           [`0x11`, FileNames],
                           [`0x12`, FileTime], [`0x13`, FileTime], [`0x14`, FileTime],
                           [`0x15`, Attributes]
FileTime             ::=   (`0x00`, bit array of TimeDefined | 0x01),
                           (`0x00`, list of Time | 0x01, DataIndex)
FileNames            ::=   (`0x00`, list of each filename | 0x01, DataIndex)
filename             ::=   Name, `0x0000`
Name                 ::=   UTF16-LE Char, [Name]
Attributes           ::=   (`0x00`, bit array of AttributesAreDefined | `0x01`),
                           (`0x00`, list of Attribute | `0x01`, DataIndex)
```

A Coder flag affect a following CoderData existence as following algorithm;

```
if (Is Complex Coder)
 {
   NUMBER ``NumInStreams``;
   NUMBER ``NumOutStreams``;
 }
 if (There Are Attributes)
 {
   NUMBER ``PropertiesSize``
   BYTE ``Properties[PropertiesSize]``
 }
}
NumBindPairs :  = ``NumOutStreamsTotal`` - 1;
for (``NumBindPairs``)
 {
   NUMBER ``InIndex``;
   NUMBER ``OutIndex``;
 }
NumPackedStreams : ``NumInStreamsTotal`` - ``NumBindPairs``;
 if (``NumPackedStreams`` > 1)
   for(``NumPackedStreams``)
   {
     NUMBER ``Index``;
   };
```

4.10 Appendix: CRC algorithm (normative)

Chunk CRCs are calculated using standard CRC methods with pre and post conditioning, as defined by ISO 3309 [ISO-3309] or ITU-T V.42 [ITU-T-V42]. The CRC polynomial employed is

```
x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1
```

The 32-bit CRC register is initialized to all 1's, and then the data from each byte is processed from the least significant bit (1) to the most significant bit (128). After all the data bytes are processed, the CRC register is inverted (its ones complement is taken). This value is transmitted (stored in the file) MSB first. For the purpose of separating into bytes and ordering, the least significant bit of the 32-bit CRC is defined to be the coefficient of the x31 term.

Practical calculation of the CRC always employs a precalculated table to greatly accelerate the computation

4.11 Appendix: Rationale

4.11.1 Byte order

It has been asked why 7-zip uses little endian byte order. It is a historical reason, that 7-zip was born as Microsoft Windows application in 1999, and its file format was a windows application format, when only little endian was used on target platform.

4.11.2 CRC32

CRC32 is a checksum.

4.11.3 Encode

Encode in this document express compressed, encrypted and/or filter data. When encoding, it should lead encoding metadata.

4.11.4 Extract

Extract in this document express decompress, decryption and/or filter data from archive.

4.11.5 UTF-16-LE

Unicode UTF-16 encoding uses 2 bytes or 4 bytes to represent Unicode characters. Because it is not one byte ordering, we need to consider endian, byte order. UTF-16-LE is a variant of UTF-16 definition which use Little-Endian for store data.

4.11.6 UTF-8

Unicode UTF-8 encoding uses a sequence of bytes, from 1 bytes to 4 bytes to represent Unicode characters. ISO 10646 defines it as 1 byts to 8 bytes encoding, so compliant implementation SHALL be able to handle 8bytes sequence and mark it as invalid.

CHAPTER

FIVE

AUTHORS

py7zr is written and maintained by Hiroshi Miura <miurahr@linux.com>

Contributors, listed alphabetically, are:

- Alan Lee – Update documentation
- Alexander Kapshuna – Fix shutil integration (#353)
- @andrebrait – Fix exception for empty 7z file (#118)
- @amarcu5 – fix error when large compressed headers (#281)
- c.foster – Default exceptions to include the exception type
- chigusa – Fix UTF-16 path parsing for extraction (#391)
- @DoNCK – Fix extraction of hidden dot files(#448)
- Jasper Lievisse Adriaanse – Update document
- Joachim Bauch – pylzma orginator
- Kazuya Fujioka – Fix zero file problem
- Kyle Altendorf – Fix multithreading problem (#82)
- Martin Larralde – Fix writef method (#397)
- Megan Leet – Fix infinite loop when extraction (#354)
- @padremayi – Fix crash on wrong crationtime in archive (#275)
- @royopa – Fix typo (#108)
- T. Yamada – Deflate64 decompression (#399)
- @Zoynels – Mmeory IO API(#111, #119)

CHAPTER

SIX

GLOSSARY

binary file
A *file object* able to read and write *bytes-like objects*. Examples of binary files are files opened in binary mode (`'rb'`, `'wb'` or `'rb+'`), `sys.stdin.buffer`, `sys.stdout.buffer`, and instances of `io.BytesIO` and `gzip.GzipFile`.

See also *text file* for a file object able to read and write `str` objects.

bytes-like object
An object that supports the *bufferobjects* and can export a C-*contiguous* buffer. This includes all `bytes`, `bytearray`, and `array.array` objects, as well as many common `memoryview` objects. Bytes-like objects can be used for various operations that work with binary data; these include compression, saving to a binary file, and sending over a socket.

Some operations need the binary data to be mutable. The documentation often refers to these as "read-write bytes-like objects". Example mutable buffer objects include `bytearray` and a `memoryview` of a `bytearray`. Other operations require the binary data to be stored in immutable objects ("read-only bytes-like objects"); examples of these include `bytes` and a `memoryview` of a `bytes` object.

contiguous
A buffer is considered contiguous exactly if it is either *C-contiguous* or *Fortran contiguous*. Zero-dimensional buffers are C and Fortran contiguous. In one-dimensional arrays, the items must be laid out in memory next to each other, in order of increasing indexes starting from zero. In multidimensional C-contiguous arrays, the last index varies the fastest when visiting items in order of memory address. However, in Fortran contiguous arrays, the first index varies the fastest.

file object
An object exposing a file-oriented API (with methods such as `read()` or `write()`) to an underlying resource. Depending on the way it was created, a file object can mediate access to a real on-disk file or to another type of storage or communication device (for example standard input/output, in-memory buffers, sockets, pipes, etc.). File objects are also called *file-like objects* or *streams*.

There are actually three categories of file objects: raw *binary files*, buffered *binary files* and *text files*. Their interfaces are defined in the `io` module. The canonical way to create a file object is by using the `open()` function.

file-like object
A synonym for *file object*.

text file
A *file object* able to read and write `str` objects. Often, a text file actually accesses a byte-oriented datastream and handles the text encoding automatically. Examples of text files are files opened in text mode (`'r'` or `'w'`), `sys.stdin`, `sys.stdout`, and instances of `io.StringIO`.

See also *binary file* for a file object able to read and write *bytes-like objects*.

path-like object
An object representing a file system path. A path-like object is either a `str` or `bytes` object representing a path,

or an object implementing the `os.PathLike` protocol. An object that supports the `os.PathLike` protocol can be converted to a `str` or `bytes` file system path by calling the `os.fspath()` function; `os.fsdecode()` and `os.fsencode()` can be used to guarantee a `str` or `bytes` result instead, respectively. Introduced by **PEP 519**.

CHAPTER

SEVEN

PY7ZR CHANGELOG

All notable changes to this project will be documented in this file.

7.1 Unreleased

7.2 v0.20.8

7.2.1 Fixed

- Detect brotli import error (#543)

7.2.2 Changed

- refactor: hardening SevenZipFile constructor (#547)
- refactor: improve type safe functions (#545)
- chore: add git export configuration (#544)

7.3 v0.20.7

7.3.1 Changed

- Support Python 3.12 (#541)

7.4 v0.20.6

7.4.1 Fixed

- fix: sanitize path when write (#525)
- fix: allow specify target path in relative path (#530)
- Avoid AttributeError on OpenBSD (#521)
- Error appending file: KeyError: 'lastwritetime' (#517)

7.4.2 Document

- Fixing a string quote in user_guide document(#524)

7.5 v0.20.5

7.5.1 Fixed

- Remove root reference from file names (#513)

7.5.2 Document

- fix typo in the readme (#510)

7.6 v0.20.4

7.6.1 Fixed

- Installation error in Cygwin (#504)

7.7 v0.20.3

7.7.1 Fixed

- Drop manual GC to improve performance when many files are handled. (#489, #490)
- CI: fix test configurations (#494) - Fix mypy error - Skip deflate64 compression/decompression test on pypy - There is an issue in dependency inflate64 library that causes SIGABORT and SIGSEGV on pypy

7.8 v0.20.2

7.8.1 Fixed

- Fix error with good path data, when detecting wrong path with new canonical_path(), and drop resolve() call on path.

7.9 v0.20.1

7.9.1 Security

- Fix sanity check for path traversal attack(#480)
- Add path checker in writef() and writestr() methods that ignores evil pass. - When pass arcname as evil path such as "../../../../tmp/evil.sh" - it raises ValueError
- Check symlink and junction is under target folder when extraction

7.10 v0.20.0

7.10.1 Added

- Support enhanced deflate compression.(#472)

7.10.2 Changed

- Bump setuptools@63 and setuptools_scm@7 (#473)
- CI: update script (#473)
- Update tox config (#473)
- Actions: change pypy version to 3.7 (#473)
- Update readthedocs.yml (#473)

7.10.3 Deprecated

- Deprecate Python 3.6 support (#473)

7.11 v0.19.0

7.11.1 Changed

- Replace deflate64(tm) decompressor to inflate64(#459)
- test: improve checks of deflate64 case(#463)

7.12 v0.18.10

7.12.1 Fixed

- Actions: fix release script to produce wheel.(#462) there is no wheel release for v0.18.5-v0.18.9

7.13 v0.18.9

7.13.1 Fixed

- Closing a SevenZipFile opened for appending, without adding a new file, raises exception (#378, #395)
- Docs: fix URL link error (#450)
- Actions: fix document compilation by installing graphviz (#450)
- Docs: fix errors and warnings on documentation.

7.13.2 Changed

- Add changelog into Documentation (#450)
- Test on python 3.11-beta (#450)
- Bump Sphinx@5.0 for Documentation (#450)
- Docs: update configuration to ignore changelog links for link check

7.14 v0.18.7

7.14.1 Fixed

- Extraction wrongly renames unix hidden dot files/directories (#448)

7.15 v0.18.6

7.15.1 Fixed

- Decompression of some LZMA+BCJ archive may abort with gegmentation fault because of a PyBCJ bug. Bump `PyBCJ@0.6.0` that fixed it. (#447)

7.15.2 Removed

- Remove in-source BCJ filter pure python code. Now it have a place in a PyBCJ project. (#447)

7.16 v0.18.5

7.16.1 Fixed

- Limit memory consumption for extraction(#430,#434,#440)
- Pyproject.toml: setuptools_scm configuration(#438)

7.16.2 Changed

- Build package with `pip wheel` with python 3.9 on Ubuntu 20.04
- Check py3.8, 3.9 and 3.10 on Azure-Pipelines CI/CD.

7.17 v0.18.4

7.17.1 Fixed

- Raise exception properly when threaded extraction(#431,#432)
- Actions: fix tox test(#433)

7.17.2 Changed

- Change pyproject.toml:license table to be text key and SPDX license name(#435, #436)

7.18 v0.18.3

7.18.1 Fixed

- ppmd: send extra byte b"0" to pyppmd.Ppmd7Decompressor, when input is exhausted, but it indicate needs_input. This is a same behavior as p7zip decoder does. (#417)
- README: fix example code(#426)

7.18.2 Changed

- Bump `PyPPMd@0.18.1` (#420,#427)
- pyproject.toml: Add project section(#428)

7.19 v0.18.1

7.19.1 Changed

- Limit dependency pyppmd to v0.17.x

7.19.2 Fixed

- Fix mypy error with mypy 0.940(#421)

7.20 v0.18.0

7.20.1 Added

- Support DEFLATE64 decompression(#399)

7.20.2 Fixed

- Docs: fix typo for readall method argument(#416)

7.20.3 Changed

- Get status down for PPMd compression/decompression(#418) PPMd decompression has a bug easily to fail decompression.

7.21 v0.17.4

7.21.1 Fixed

- When extracting and target archive compressed with unsupported LZMA2+BCJ2, py7zr raise unexpected exception. Fix to raise better exception message

7.21.2 Changed

- docs: Add explanation of empty file specification

7.22 v0.17.3

7.22.1 Security

- Check against directory traversal attack by file pathes in archive (#406,#407)

7.23 v0.17.2

7.23.1 Fixed

- writef method detect wrong size of data(#397)

7.23.2 Changed

- Improve callback object check and error message(#387)

7.24 v0.17.1

7.24.1 Fixed

- Allow 7zAES+LZMA2+BCJ combination for compression(#392)
- Argument error when raising UnsupportedCompressionMethodError(#394)
- Detect memory leak in test and fix some leaks(#388)
- Fix filename and property decode in UTF-16(#391)

7.24.2 Changed

- Azure: use `macos@10.15` for test(#389)

7.25 v0.17.0

7.25.1 Fixed

- Extraction: overwrite a symbolic link sometimes failed(#383)
- Allow creation of archive without any write call(#369,#372)
- Type check configuration update (#384)
- Adjust for type check errors (#384)

7.26 v0.16.4

7.26.1 Fixed

- Win32 file namespace convention doesn't work on Cygwin(#380,#381)
- Win32 file namespace convention doesn't work for network path(#380)

7.27 v0.16.3

7.27.1 Fixed

- Reduce memory consumptions and fix memory_error on 32bit python (#370,#373,#374,#375)

7.27.2 Added

- Add CI test for python 3.10 (#371)

7.28 v0.16.2

7.28.1 Added

- Bundle type hint data
- README: Add conda recipe(#342)

7.28.2 Changed

- Use PyBCJ instead of bcj-cffi.(#368)
- Docs: change recommended python versions
- CI: benchmark on python 3.10
- Test expectation for python 3.10 change
- Improve exceptions and error messages
- Docs: add description of ArchiveInfo class
- Docs: fix typo on shutil integration(#353)
- Bump pyzstd@0.15.0
- Bump pyppmd@0.17.0

7.28.3 Fixed

- Docs: specification error of signature header data types.
- Fix infinite loop in extract(#354)

7.29 v0.16.1

7.29.1 Added

- type hint for mypy

7.30 v0.16.0

7.30.1 Added

- Add Brotli compression.
- CI: Test on AArch64.

7.30.2 Changed

- CLI: support multi-volume archive without making temporary file(#311)
- Filter parameter: PPMd: mem is now accept int or "<val>{m|k|b}" as same as 7-zip command line option. int value is recognized as "1 << val" ie. 24 means 4MB.
- Dependency: PyPPMd v0.14.0+
- Dependency PyCryptodome to PyCryptodomex that changes package name from PyCrypto to PyCryptodome(#334)

CHAPTER

EIGHT

PREVIOUS CHANGES

8.1 v0.15.2

8.1.1 Added

- CLI: create sub-command(c) has -P or –password option.(#332)

8.1.2 Fixed

- Fix not to produce directory when memory extraction mode.(#323)

8.1.3 Changed

- Use PyPPMd v0.12.1 or later for ppmd compression instead of ppmd-cffi(#322)
- Add minimum version requirement for PyCryptodome (#329)
- Bump setuptools_scm @6.0.1

8.2 v0.15.1

8.2.1 Changed

- Update release automation script.
- Bump ppmd-cffi and bcj-cffi versions(#320)

8.3 v0.15.0

8.3.1 Added

- Add option to specify multiprocessing instead of multi-threading. (#306)

8.3.2 Changed

- Change Property Borg class to constant class(#319)
- Reformat whole code with black.
- Merge pyzstdfilter into compressor.py.
- Lint codes by flake8/black.

8.3.3 Fixed

- README: description of dependencies.
- ZStandard decompression on PyPy3

8.4 v0.14.1

8.4.1 Fixed

- Fix of empty file archive(#305,#310)

8.5 v0.14.0

8.5.1 Added

- Introduce writed() method that accept dict[name, BinaryIO](#302)

8.5.2 Changed

- READ_BLOCKSIZE configurable on constructor(#307)
- Use pyzstd for zstandard algorithm on CPython(#304)
- Use bcj-cffi library for lzma+bcj performance(#303)
- CLI: Fix getting module_name on 3.6.13(#308)

8.6 v0.13.0

8.6.1 Added

- Add writestr() and writef() methods in SevenZipFile class.(#290,#293)
- Add benchmark tests for compression algorithms(#295)
- Track benchmark results on Github issue(#296)

8.6.2 Changed

- Refactoring BCF Filter classes, and move to individual module.(#292)

8.7 v0.12.0

8.7.1 Changed

- PPMd and ZStandard is now one of default algorithms(#289)
- Increment copyright year

8.7.2 Fixed

- Crash when append files to an empty files archive(#286)

8.8 v0.11.3

8.8.1 Fixed

- Fix test failure when running on pypi source(#279)

8.8.2 Security

- Drop issue_218.7z test data wihch is reported a blackmoon trojan(#285)

8.9 v0.11.1

8.9.1 Changed

- Improve BCJ filter performance with LZMA1, ZStd compressions.

8.9.2 Fixed

- Fix to allow writing encrypted header(#280)
- Avoid crash when creationtime is wrong or Unix epoch. (#275,#276)

8.10 v0.11.0

8.10.1 Changed

- PPMd: Use stream encoder/decoder instead of buffered one.
- PPMd: Use ppmd-cffi@v0.3.1 and later.(#268)

8.10.2 Added

- PPMd compression/decompression support.(#255)
- New API to set methods to set header encode mode, encode or encrypted.(#259)
- Support Python 3.9.(#261)
- Support arm64/aarch64 architecture on Linux.(#262)

8.10.3 Fixed

- Append mode cause error when target archive use LZMA2+BCJ.(#266)
- Fix zstandard compression/decompression.(#258)

8.10.4 Deprecated

- Drop support for python 3.5 which become end-of-line in Sept. 2020.

8.11 v0.10.1

8.11.1 Fixed

- Fix exception when reading header which size is larger than buffer size (#252)

8.12 v0.10.0

8.12.1 Added

- Compatibility test with python-libarchive-c/libarchive for compression(#247)
- Document: express how to handle multi-volume archive (#243)
- SevenZipFile.needs_password() method.(#208, #235)
- CLI: Support append mode command line.(#228)
- Support "APPEND" mode. User can open SevenZipFile() class with mode='a' (#227)

8.12.2 Changed

- Calculate CRC32 of header without re-reading header from disk again.(#245)
- read(), extract(): improve performance when specifying parts of archived file, by skipping rest of arcvhive when target file has extracted.(#239,#242)
- read(), extract(): improve performance when specifying parts of archived file, by not running threads for unused compression blocks(folders).(#239,#242)
- docs: improve API documentation.(#244)
- setup: set minimum required python version as >=3.5
- Compression will be happened when call write() not close() (#222, #226)
- Handle file read/write in SevenZipCompressor/Decompressor class (#213)

8.12.3 Fixed

- Fix BCJ(x86) filter code with a missing logic which cause extraction error for certain data. (#249, #250)
- Raise PasswordRequired when encrypted header without passing password (#234, #237)
- CLI: don't raise exception when password is wrong or not given.(#229)
- Fix specification typo.
- Catch exception in threading extraction(#218,#219)

8.13 v0.9.2

8.13.1 Changed

- Utilize max_length argument for each decompressor.(#210, #211)
- Change READ_BUFFER_SIZE 32768 for python 3.7.5 and before.
- Extend Buffer size when necessary.(#209)

8.14 v0.9.1

8.14.1 Changed

- Improve DecompressionChain.decompress() logics.(#207)

8.14.2 Fixed

- Fix BCJ filter for decompression that can cause infinite loop or wrong output.(#204,#205,#206)

8.15 v0.9.0

8.15.1 Added

- BCJ Decoder/Encoder written by python.(#198, #199)
- Support Bzip2, Defalte + BCJ(X86, PPC, ARM, ARMT, SPARC) (#199)
- Add Copy method as an extraction only support.(#184)

8.15.2 Changed

- Use large(1MB) read blocksize for Python 3.7.5 and later and PyPy 7.2.0 and later.
- Set ZStandard compression as unsupported because of a bug with unknown reason.(#198)
- Manage compression methods to handle whether decompressor requires coder['property'] or not.

8.15.3 Fixed

- Significantly improve decompress performance which is as same speed as v0.7.*. by updating buffer handling.
- Fix decompression max_size to pass lzma module. Now it is as same as out_remaining.
- Support LZMA+BCJ(X86, PPC, ARM, ARMT, SPARC) with alternative BCJ filter.(#198, #199)
- Fix packinfo crc read and write (#187, #189)
- Accept archive which Method ID is NULL(size=0)(#181, #182)
- CLI: Does not crash when trying extract archive which use unsupported method(#183)

8.16 v0.8.0

8.16.1 Added

- test: add test for #178 bug report the case of LZMA+BCJ as xfails.
- File format specification: add ISO/IEC standard style specification document.
- Support extra methods for archiveinfo() method.(#150)
- test: unit tests for Sparc, ARMT and IA64 filters.
- Support for PPC and ARM filters.
- Support encryption(#145)
- Export supported filter constants, such as FILTER_ZSTD(#145)

8.16.2 Changed

- Improve README, documents and specifications.
- Update password handling and drop get_password() helper (#162)
- Enable encoded header and add more test with 7zip compatibility.(#164)
- Refactoring SevenZipFile class internals. (#160)
- Refactoring classes in compressor module. (#161)
- Add 'packinfo.crcs' field digests data when creating archive.(#157) It help checking archive integrity without extraction.
- CLI: help option to show py7zr version and python version.
- Use importlib for performance improvement instead of pkg_resources module.
- Documents: additional methods, filter examples.
- CI configurations: Manage coverage with Coveralls.
- Refactoring decompression classes to handle data precisely with folder.unpacksizes(#146)
- Default compression mode is LZMA2+BCJ which is as same as 7zip and p7zip(#145)
- Enhance encryption strength, IV is now 16 bytes, and generated with cryptodom.random module.(#145)
- Refactoring compression algorythm related modules.

8.16.3 Fixed

- Now return correct header size by archiveinfo() method.(#169)
- Disable adding CRC for encoded header packinfo.(#164)
- Fix password leak/overwrite among SevenZipFile objects in a process.(#159) This can cause decryption error or encryption with unintended password.
- Release password on close()
- SevenZipFile.test() method now working properly. (#155)
- Fix extraction error on python 3.5.(#151)
- Support combination of filters(#145)
- Compression of Delta, BZip2, ZStandard, and Deflate(#145)
- Fix archived head by multiple filter specified.
- Fix delta filter.
- Working with BCJ filter.
- Fix archiveinfo to provide proper names.

8.16.4 Removed

- test: Drop some test case with large files.
- Drop ArchiveProperty class: A field has already deprecated or not used.(#170)
- Drop AntiFile property: a property has already deprecated or not used.
- remove final_header definition.

8.17 v0.7.3

8.17.1 Added

- Support for encrypted header (#139, #140)

8.17.2 Changed

- Fix CRC32 check and introduce test and testzip methods (#138)

8.17.3 Fixed

- Allow decryption of data which is encrypted without any compression.(#140)

8.18 v0.7.2

8.18.1 Added

- CLI: '-v {size}[b|k|m|g]' multi volume creation option.

8.19 v0.7.1

8.19.1 Changed

- Decryption: performance improvement. Introduce helpers.calculate_key3(), which utilize list comprehension expression, bytes generation with join(). It reduces a number of calls of hash library and improve decryption performance.

8.19.2 Fixed

- Fix overwrite behavior of symbolic link which may break linked contents.

8.20 v0.7.0

8.20.1 Added

- Support dereference option of SevenZipFile class. (#131) If dereference is False, add symbolic and hard links to the archive. If it is True, add the content of the target files to the archive. This has no effect on systems that do not support symbolic links.
- Introduce progress callback mechanism (#130)
- Support memory API.(#111, #119) Introduce read(filter) and readall() method for SevenZipFile class.
- Support ZStandard codec compression algorithm for extraction.(#124, #125)

8.20.2 Changed

- Extraction: Unlink output file if exist when it become a symbolic link. When overwrite extracted files and there are symlinks, it may cause an unexpected result. Unlinking it may help it.
- CLI: add –verbose option for extraction
- win32: update win32compat
- Drop pywin32 dependency(#120)
- Introduce internal win32compat.py
- Archive: Looking for symbolic link object in the archived list, and if found, record as relative link.(#112, #113, #122)

8.20.3 Fixed

- Fix archiveinfo() for 7zAES archives
- Release variables when close() (#129)
- Support extraction of file onto a place where path length is > 260 bytes on Windows 10, Windows Server 2016R2 and later. (Windows Vista, 7 and Windows Server 2012 still have a limitation of path length as a OS spec)(#116, #126)

8.20.4 Removed

- Revmoed requirements.txt. When you want to install dependencies for development you can do it with 'pip install -e path/to/py7zr_project'

8.21 v0.6

8.21.1 Added

- Test: SevenZipFile.archiveinfo() for various archives.
- Test: extraction of LZMA+BCJ archive become fails as marked known issue.
- Support deflate decompression method.
- Introduce context manager for SevenZipFile (#95)
- Test: add benchmarking test.
- Add concurrent extraction test.
- Add remote data test for general application test.
- Add class for multi volume header.
- Add readlink helper function for windows.
- Test: download and extract test case as a show case.
- setup.cfg: add entry-point configuration.
- Support filtering a target of extracted files from archive (#64)
- Support decryption (#55)
- Add release note automation workflow with Github actions.
- COPY decompression method.(#61)

8.21.2 Changed

- Update documents and README about supported algorithms.
- Re-enable coverage report.
- Refactoring SevenZipFile._write_archive() method to move core chunk into compression module Worker.archive() method.
- Update calculate_key helper to improve performance.
- Introduce zero-copy buffer helper.
- **Change decompressor class interface**
  - change max_length type to int and defualt to -1.
- Update decryption function to improve performance.
- SevenZipFile(file-object, 'r') now can run extract() well even unlink before extract().
- Concurrency strategy: change to threading instead of multiprocessing. (#92)
- Release process is done by Github Actions
- Temporary disable to measure coverage, which is not working with threading.
- Tox: now pass PYTEST_ADDOPTS environment variable.
- extract: decompression is done as another process in default.
- extract: default multiprocessing mode is spawn

- extract: single process mode for password protected archive.
- Use spawn multiprocessing mode for all platforms.
- Use self context for multiprocessing.
- Concurrency implementation changes to use multiprocessing.Process() instead of concurrency.futures to avoid freeze or deadlock with application usage of it.(#70)
- Stop checking coverage because coverage.py > 5.0.0 produce error when multiprocessing.Process() usage.
- Drop handlers, NullHandler, BufferHnalder, and FileHander.

8.21.3 Fixed

- Fix SevenZipFIle.archiveinfo() crash for LZMA+BCJ archive.(#100)
- Fix SevenZipFile.test() method defeated from v0.6b2 (#103)
- Fix SevenZipFile.solid() method to return proper value. (#72,#97)
- Fix README example for extraction option.
- Some of decryption of encrypted archive fails.(#75)
- Make pywin32 a regular runtime dependency
- Build with pep517 utility.
- Fix race condition for changing current working directory of caller, which cause failures in multithreading.(#80,#82)
- extract: catch UnsupportedMethod exception properly when multiprocessing.
- Fixed extraction of 7zip file with BZip2 algorithm.(#66)
- Fix symbolic link extraction with relative path target directory.(#67)
- Fix retrieving Folder header information logics for codecs.(#62)

8.21.4 Security

- CLI: Use 'getpass' standard library to input password.(#59)

8.21.5 Removed

- Static py7zr binary. Now it is generated by python installer.
- Test symlink on windows.(#60)

8.22 v0.5

Support making a 7zip archive.

8.22.1 Added

- Support for compression and archiving.
- Support encoded(compressed) header and set as default.(#39)
- SevenZipFile: accept pathlib.Path as a file argument.
- Unit test: read and write UTF-16LE string for filename.
- Support for shutil.register_archive_format() and shutil.make_archive() by exposing pack_7zarchive()
- Support custom filters for compression.

8.22.2 Changed

- Update documents.

8.22.3 Fixed

- Fix extraction of archive which has zero size files and directories(#54).
- Revert zero size file logic(#47).
- Revert zero size file logic which break extraction by 7zip.
- Support for making archive with zero size files(#47).
- Produced broken archive when target has many directorires(#48).
- Reduce test warnings, fix annotations.
- Fix coverage error on test.
- Support for making archive with symbolic links.
- Fix write logics (#42)
- Fix read FilesInfo block.
- Skip rare case when directory already exist, that can happen multiple process working in same working directory.
- Write: Produce a good archive file for multiple target files.
- SignatureHeader function: write nextheaderofs and nextheadersize as real_uint64.
- docs: description of start header structure.

8.22.4 Removed

- Drop py7zr.properties.FileAttributes; please use stat.FILE_ATTRIBUTES_*

8.22.5 Changed

- Test: Use tmp_path fixture which is pytest default one.
- Move setuptools configurations in setup.py into setup.cfg.

8.23 v0.4

8.23.1 Added

- Support for pypy3 (pypy3.5-7.0) and later(pypy3.6-7.1 or later).
- unit test for NullHandler, BufferHandler, FileHandler.
- Update document to add 7zformat descriptions.

8.23.2 Changed

- NullHandler, BufferHandler, FileHandler: open() now takes mode argument.
- Upper limit of max_length of decompress() call is now io.DEFAULT_BUFFER_SIZE. - PyPy issue: PyPy3-3088
- Drop padding logic introduced in v0.3.5 that may be cuased by python core bug, when max_length > io.DEFAULT_BUFFER_SIZE. - PyPy Issue: PyPy3-3090 - bpo-21872: https://bugs.python.org/issue21872 - Fix: https://github.com/python/cpython/pull/14048
- **Remove print functions from API and moves CLI**
  - API should not output anything other than error message. * Introduce FileInfo class to represent file attributes inside archive. * Introduce ArchiveInfo class to represent archive attributes. * provide archiveinfo() method to provide ArchiveInfo object. * now list() method returns List[FileInfo]
  - Every print things moves to Cli class.
- Update tests according to API change.
- Update documents to refrect API changes.

8.23.3 Fixed

- Update README to indicate supported python version as 3.5 and later, pypy3 7.1 and later.

8.24 v0.3.5

8.24.1 Changed

- Use seek&truncate for padding trailer if needed.

8.25 v0.3.4

8.25.1 Added

- Docs: class diagram, design note, 7z formats and presentations.
- Test for a target includes padding file.

8.25.2 Changed

- Test file package naming.

8.25.3 Fixed

- Fix infinite loop when archive file need padding data for extraction.

8.26 v0.3.3

8.26.1 Added

- Add test for zerofile with multi-foler archive.

8.26.2 Fixed

- Fix zerofile extraction error with multithread mode(#24, thanks @Arten013)

8.27 v0.3.2

8.27.1 Added

- typing hints
- CI test with mypy
- Unit test: SignatureHeader.write() method.
- Unit test: unknown mode for SevenZipFile constructor.
- Unit test: SevenZipFile.write() method.

8.27.2 Changed

- Conditional priority not likely to be external in header.
- Refactoring read_uint64().

8.27.3 Fixed

- SignatureHeader.write(): fix exception to write 7zip version.

8.28 v0.3.1

8.28.1 Added

- CLI i subcommand: show codec information.
- Decompression performance test as regression test.
- Add more unit test for helper functions.

8.28.2 Changed

- List subcommand now do not show compressed file size in solid compression. This is as same behavior as p7zip command.
- Merge io.py into archiveinfo.py
- Drop internal intermediate queue, which is not used.

8.28.3 Fixed

- Always overwrite when archive has multiple file with same name.

8.29 v0.3

8.29.1 Added

- Add some code related to support write feature(wip).
- Static check for import order in python sources and MANIFEST.in

8.29.2 Changed

- Concurrent decompression with threading when an archive is in multi folder compression.
- Pytest configurations are set in tox.ini

8.29.3 Fixed

- Package now has test code and data.

8.30 v0.2.0

8.30.1 Fixed

- Detect race condition on os.mkdir

8.31 v0.1.6

8.31.1 Fixed

- Wrong file size when lzma+bcj compression.

8.32 v0.1.5

8.32.1 Fixed

- Suppress warning: not dequeue from queue length 0

8.33 v0.1.4

8.33.1 Changed

- When a directory exist for target, do not raise error, and when out of it raise exception
- Refactoring FileArchivesList and FileArchive classes.

8.34 v0.1.3

8.34.1 Changed

- When a directory exist for target, do not raise error, and when out of it raise exception

8.35 v0.1.2

8.35.1 Changed

- Refactoring CLI with cli package and class.

8.35.2 Fixed

- Archive with zero size file cause exception with file not found error(#4).

8.35.3 Removed

- Drop unused code chunks.
- Drop Digests class and related unit test.

8.36 v0.1.1

8.36.1 Added

- Add write(), close() and testzip() dummy methods which raises NotImplementedError.
- Add more unit tests for write functions.

8.36.2 Fixed

- Fix Sphinx error in documentation.
- SevenZipFile: Check mode before touch file.
- Fix write_boolean() when array size is over 8.
- Fix write_uint64() and read_uint64().

8.37 v0.1.0

8.37.1 Added

- Introduce compression package.
- Introduce SevenZipCompressor class.
- Add write() method for each header class.
- Add tests for write methods.
- Add method for registering shutil.

8.37.2 Changed

- Each header classes has __slots__ definitions for speed and memory optimization.
- Rename to 'io' package from 'archiveio'
- Each header classes has classmethod 'retrieve' and constructor does not reading a archive file anymore.
- Change to internalize _read() method for each header classes.
- get_decompressor() method now become SevenZipDecompressor class.
- Each header classes initializes members to None in constructor.
- Method definitions map become an internal member of SevenZipDecompressor or SevenZipCompressor class.
- Add test package compress

8.37.3 Fixed

- Fix ArchiveProperties read function.

8.38 v0.0.8

8.38.1 Added

- Test for CLI.

8.38.2 Changed

- Improve main function.
- Improve tests, checks outputs with sha256

8.39 v0.0.7

8.39.1 Added

- CI test on AppVeyor.

8.39.2 Changed

- Worker class refactoring.

8.39.3 Fixed

- Fix test cases: bugzilla_16 and github_14.
- Test: set timezone to UTC on Unix and do nothing on Windows.

8.40 v0.0.6

8.40.1 Fixed

- Fix too many file descriptors opened error.

8.41 v0.0.5

8.41.1 Changed

- Test: check sha256 for extracted files

8.41.2 Fixed

- Fix decompressiong archive with LZMA2 and BCJ method
- Fix decompressing multi block archive
- Fix file mode on unix/linux.

8.42 v0.0.4

8.42.1 Added

- Set file modes for extracted files.
- More unit test.

8.42.2 Changed

- Travis-CI test on python 3.7.

8.42.3 Fixed

- Fix to set extracted files timestamp as same as archived.

8.43 v0.0.3

8.43.1 Added

- PyPi package index.

8.43.2 Fixed

- setup: set universal = 0 because only python 3 is supported.

8.44 v0.0.2

8.44.1 Changed

- refactoring all the code.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

P

R

S

T

U

W

X

Z