Zip (file format)

From Wikipedia, the free encyclopedia

ZIP is an archive file format that supports lossless data compression. A .ZIP file may contain one or more files or directories that may have been compressed. The .ZIP file format permits a number of compression algorithms, though DEFLATE is the most common. This format was originally created in 1989 by Phil Katz, and was first implemented in PKWARE, Inc.'s PKZIP utility,[2] as a replacement for the previous ARC compression format by Thom Henderson. The .ZIP format is now supported by many software utilities other than PKZIP. Microsoft has included built-in .ZIP support (under the name "compressed folders") in versions of Microsoft Windows since 1998. Apple has included built-in .ZIP support in Mac OS X 10.3 (via BOMArchiveHelper, now Archive Utility) and later. Most free operating systems have built in support for .ZIP in similar manners to Windows and Mac OS X.

ZIP file format

Filename extension	`.zip`, `.zipx` (newer compression algorithms)
Internet media type	`application/zip`[1]
Uniform Type Identifier (UTI)	com.pkware.zip-archive
Magic number	none, though `PK\003\004` , `PK\005\006` (empty archive), or `PK\007\008` (spanned archive) are common.
Developed by	Phil Katz, PKWARE, Inc.
Initial release	1989
Latest release	6.3.4 (1 October 2014)
Type of format	Data compression
Extended to	JAR (EAR, RAR (Java), WAR) Office Open XML (Microsoft) Open Packaging Conventions OpenDocument (ODF) XPI (Mozilla extensions)
Standard	APPNOTE (http://www.pkware.com/documents/casestudies/APPNOTE.TXT) from PKWARE ISO/IEC 21320-1:2015 (a subset of ZIP file format 6.3.3) (http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=60101)

.ZIP files generally use the file extensions ".zip" or ".ZIP" and the MIME media type `application/zip`.[1] ZIP is used as a base file format by many programs, usually under a different name. When navigating a file system via a user interface, graphical icons representing .ZIP files often appear as a document or other object prominently featuring a zipper.

Contents

History

The .ZIP file format was created by Phil Katz of PKWARE. He created the format after his company had lawsuits filed against him by Systems Enhancement Associates (SEA) claiming that his archiving products were derivatives of SEA's ARC archiving system. The name "zip" (meaning "move at high speed") was suggested by Katz's friend, Robert Mahoney. They wanted to imply that their product would be faster than ARC and other compression formats of the time. The earliest known version of *.ZIP File Format Specification* was first published as part of PKZIP 0.9 package under the file APPNOTE.TXT in 1989.

The .ZIP file format was released into the public domain.[3][4][5][6][7]

Version history

The .ZIP File Format Specification has its own version number, which does not necessarily correspond to the version numbers for the PKZIP tool, especially with PKZIP 6 or later. At various times, PKWARE has added preliminary features that allow PKZIP products to extract archives using advanced features, but PKZIP products that create such archives are not made available until the next major release. Other companies or organizations support the PKWARE specifications at their own pace.

The .ZIP file format specification is formally named "APPNOTE - .ZIP File Format Specification" and it is published on the PKWARE.com website since the late 1990s.[8] Several versions of the specification were not published. Specifications of some features such as BZIP2 compression, strong encryption specification and others were published by PKWARE a few years after their creation. The URL of the online specification was changed several times on the PKWARE website.

A summary of key advances in various versions of the PKWARE specification:

- 2.0: (1993)[1] File entries can be compressed with DEFLATE and use traditional PKWARE encryption.
- 2.1: (1996) Deflate64 compression
- 4.5: (2001)[9] Documented 64-bit zip format.
- 4.6: (2001) BZIP2 compression (not published online until the publication of APPNOTE 5.2)
- 5.0: (2002) DES, Triple DES, RC2, RC4 supported for encryption (not published online until the publication of APPNOTE 5.2)
- 5.2: (2003)[10][11] AES encryption support (defined in APPNOTE 5.1 that was not published online), corrected version of RC2-64 supported for encryption.
- 6.1: (2004)[12] Documented certificate storage.
- 6.2.0: (2004)[13] Documented Central Directory Encryption.
- 6.3.0: (2006)[14] Documented Unicode (UTF-8) filename storage. Expanded list of supported hash, compression (LZMA, PPMd+), encryption algorithms.
- 6.3.1: (2007)[15] Corrected standard hash values for SHA-256/384/512.
- 6.3.2: (2007)[16] Documented compression method 97 (WavPack).
- 6.3.3: (2012)[17] Document formatting changes to facilitate referencing the PKWARE Application Note from other standards using methods such as the JTC 1 Referencing Explanatory Report (RER) as directed by JTC 1/SC 34 N 1621.
- 6.3.4: (2014)[18] Updates the PKWARE, Inc. office address.

WinZip, starting with version 12.1, uses the extension `.zipx` for .ZIP files that use compression methods newer than DEFLATE; specifically, methods BZip, LZMA, PPMd, Jpeg and Wavpack. The last 2 are applied to appropriate file types when "Best method" compression is selected.[19][20]

Standardization

In April 2010, ISO/IEC JTC 1 initiated a ballot to determine whether a project should be initiated to create an ISO/IEC International Standard format compatible with .ZIP.[21] The proposed project, entitled *Document Packaging*, envisaged a .ZIP-compatible 'minimal compressed archive format' suitable for use with a number of existing standards including OpenDocument, Office Open XML and EPUB.

In July 2010, the ballot for initiating this project failed to pass an international vote and was rejected through ISO/IEC JTC 1/SC 34 N 1461. Comments against this project cited the recognition that an existing published work on the .ZIP format has been in existence for over 18 years in the form of the PKWARE APPNOTE, recommending instead "for JTC 1 to approve the ZIP Application Note as a Referenced Specification (RS) per Annex N of the currently published JTC 1 Directives".

This ballot did approve a request for the formation of a study period for the purpose of seeking wider input regarding this core technology. The study period, which began in October 2010, brought together a number of international experts to discuss using .ZIP within international standards. In March, 2011 this group presented to JTC 1 a new recommendation on how to incorporate .ZIP within international standards.

Acknowledging the broad interoperability that the .ZIP format has achieved the study group concluded in their recommendation that "the best way to achieve our technical objectives is to have PKWARE continue its maintenance of the ZIP Application Note." The recommendations drafted by this study group were presented for balloting as ISO/IEC JTC 1/SC 34 N 1621[22] in July, 2011 and was approved by an international vote.

Proposal N 1621 directs international standards that use .ZIP to "not duplicate or contradict the provisions of PKWARE's ZIP Application Note, [and to] reference the ZIP Application Note's capabilities via an external normative reference to the latest version of the ZIP Application Note." Standards using .ZIP should include a JTC 1 Referencing Explanatory Report (RER) when referencing the PKWARE Application Note.

A provision of N 1621 included an option for drafting a profile standard for referencing .ZIP. This profile could be used by other international standards that use .ZIP to avoid having to write their own RER document where similar use of .ZIP may exist. At this time, no standards that use .ZIP have requested this profile.

Document Container File - Part 1

In October 2015, after the standardization process, the standard was published under the name *ISO/IEC 21320-1:2015 - Information technology -- Document Container File -- Part 1: Core*.[23] ISO/IEC 21320-1:2015 normatively references the PKWARE Zip File Format Specification version 6.3.3. The format is a compatible profile of that defined by the Zip Application Note. A preparatory draft of the text is available as a public document.[24]

ISO/IEC 21320-1:2015 requires the following main restrictions of the ZIP file format:

- Files in ZIP archives may only be stored uncompressed, or using the "deflate" compression (i.e. compression method may contain the value "0" - stored or "8" - deflated).
- The encryption features are prohibited.
- The digital signature features are prohibited.
- The "patched data" features are prohibited.
- Archives may not span multiple volumes or be segmented.

Design

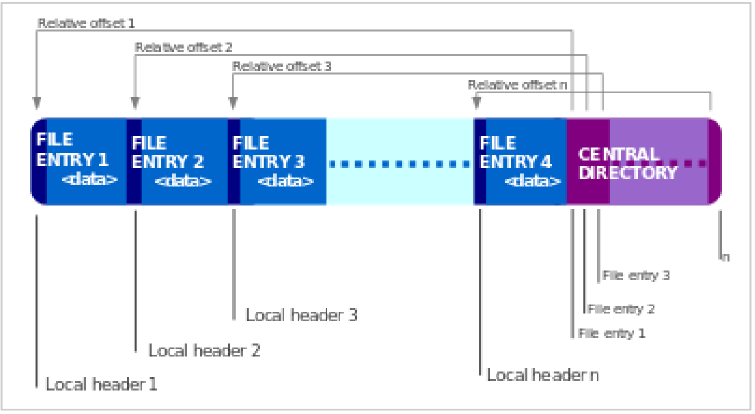
.ZIP files are archives that store multiple files. .ZIP allows contained files to be compressed using many different methods, as well as simply storing a file without compressing it. Each file is stored separately, allowing different files in the same archive to be compressed using different methods. Because the files in a .ZIP archive are compressed individually it is possible to extract them, or add new ones, without applying compression or decompression to the entire archive. This contrasts with the format of compressed tar files, for which such random-access processing is not easily possible.

A directory is placed at the end of a .ZIP file. This identifies what files are in the .ZIP and identifies where in the .ZIP that file is located. This allows .ZIP readers to load the list of files without reading the entire .ZIP archive. .ZIP archives can also include extra data that is not related to the .ZIP archive. This allows for a .ZIP archive to be made into a self-extracting archive (application that decompresses its contained data), by prepending the program code to a .ZIP archive and marking the file as executable. Storing the catalog at the end also makes possible hiding a zipped file by appending it to an innocuous file, such as a GIF image file.

The .ZIP format uses a 32-bit CRC algorithm and includes two copies of the directory structure of the archive to provide greater protection against data loss.

Structure

A .ZIP file is correctly identified by the presence of an *end of central directory record* which is located at the end of the archive structure in order to allow the easy appending of new files. If the end of central directory record indicates a non-empty archive, the name of each file or directory within the archive should be specified in a *central directory* entry, along with other metadata about the entry, and an offset into the .ZIP file, pointing to the actual entry data. This allows a file listing of the archive to be performed relatively quickly, as the entire archive does not have to be read to see the list of files. The entries within the .ZIP file also include this information, for redundancy, in a *local file header*. Because zip files may be appended to, only files specified in the central directory at the end of the file are valid. Scanning a ZIP file for local file headers is invalid (except in the case of corrupted archives), as the central directory may declare that some files have been deleted and other files have been updated.



ZIP-64 Internal Layout

For example, we may start with a .ZIP file that contains files A, B and C. File B is then deleted and C updated. This may be achieved by just appending a new file C to the end of the original ZIP file and adding a new central directory that only lists file A and the new file C. When ZIP was first designed, transferring files by floppy disk was common, yet writing to disks was very time consuming. If you had a large zip file, possibly spanning multiple disks, and only needed to update a few files, rather than reading and re-writing all the files, it would be substantially faster to just read the old central directory, append the new files then append an updated central directory.

The order of the file entries in the central directory need not coincide with the order of file entries in the archive.

Each entry stored in a ZIP archive is introduced by a *local file header* with information about the file such as the comment, file size and file name, followed by optional "extra" data fields, and then the possibly compressed, possibly encrypted file data. The "Extra" data fields are the key to the extensibility of the .ZIP format. "Extra" fields are exploited to support the ZIP64 format, WinZip-compatible AES encryption, file attributes, and higher-resolution NTFS or Unix file timestamps. Other extensions are possible via the "Extra" field. .ZIP tools are required by the specification to ignore Extra fields they do not recognize.

The .ZIP format uses specific 4-byte "signatures" to denote the various structures in the file. Each file entry is marked by a specific signature. The end of central directory record is indicated with its specific signature, and each entry in the central directory starts with the 4-byte *central file header signature*.

There is no BOF or EOF marker in the .ZIP specification. Conventionally the first thing in a .ZIP file is a .ZIP entry, which can be identified easily by its *local file header signature*. However, this is not necessarily the case, as this not required by the .ZIP specification - most notably, a self-extracting archive will begin with an executable file header.

Tools that correctly read .ZIP archives must scan for the end of central directory record signature, and then, as appropriate, the other, indicated, central directory records. They must not scan for entries from the top of the ZIP file, because only the central directory specifies where a file chunk starts. Scanning could lead to false positives, as the format does not forbid other data to be between chunks, nor file data streams from containing such signatures. However, tools that attempt to recover data from damaged .ZIP archives will most likely scan the archive for local file header signatures; this is made more difficult by the fact that the compressed size of a file chunk may be stored after the file chunk, making sequential processing difficult.

Most of the signatures end with the short integer 0x4b50, which is stored in little-endian ordering. Viewed as an ASCII string this reads "PK", the initials of the inventor Phil Katz. Thus, when a .ZIP file is viewed in a text editor the first two bytes of the file are usually "PK". (DOS, OS/2 and Windows self-extracting ZIPs have an EXE before the ZIP so start with "MZ"; self-extracting ZIPs for other operating systems may similarly be preceded by executable code for extracting the archive's content on that platform.)

The .ZIP specification also supports spreading archives across multiple filesystem files. Originally intended for storage of large .ZIP files across multiple floppy disks, this feature is now used for sending .ZIP archives in parts over email, or over other transports or removable media.

The FAT filesystem of DOS has a timestamp resolution of only two seconds; .ZIP file records mimic this. As a result, the built-in timestamp resolution of files in a .ZIP archive is only two seconds, though extra fields can be used to store more precise timestamps. The .ZIP format has no notion of time zone, so timestamps are only meaningful if it is known what time zone they were created in.

In September 2007, PKWARE released a revision of the .ZIP specification providing for the storage of file names using UTF-8, finally adding Unicode compatibility to .ZIP.[25]

File headers

All multi-byte values in the header are stored in little-endian byte order. All length fields count the length in bytes.

Local file header

Offset	Bytes	Description[25]
0	4	Local file header signature = 0x04034b50 (read as a little-endian number)
4	2	Version needed to extract (minimum)
6	2	General purpose bit flag
8	2	Compression method
10	2	File last modification time
12	2	File last modification date
14	4	CRC-32
18	4	Compressed size
22	4	Uncompressed size
26	2	File name length (*n*)
28	2	Extra field length (*m*)
30	*n*	File name
30+*n*	*m*	Extra field

The extra field contains a variety of optional data such as OS-specific attributes. It is divided into chunks, each with a 16-bit ID code and a 16-bit length.

This is immediately followed by the compressed data.

If the bit at offset 3 (0x08) of the general-purpose flags field is set, then the CRC-32 and file sizes are not known when the header is written. The fields in the local header are filled with zero, and the CRC-32 and size are appended in a 12-byte structure (optionally preceded by a 4-byte signature) immediately after the compressed data:

Data descriptor

Offset	Bytes	Description[25]
0	0/4	*Optional* data descriptor signature = 0x08074b50
0/4	4	CRC-32
4/8	4	Compressed size
8/12	4	Uncompressed size

The central directory entry is an expanded form of the local header:

Central directory file header

Offset	Bytes	Description[25]
0	4	Central directory file header signature = 0x02014b50
4	2	Version made by
6	2	Version needed to extract (minimum)
8	2	General purpose bit flag
10	2	Compression method
12	2	File last modification time
14	2	File last modification date
16	4	CRC-32
20	4	Compressed size
24	4	Uncompressed size
28	2	File name length (*n*)
30	2	Extra field length (*m*)
32	2	File comment length (*k*)
34	2	Disk number where file starts
36	2	Internal file attributes
38	4	External file attributes
42	4	Relative offset of local file header. This is the number of bytes between the start of the first disk on which the file occurs, and the start of the local file header. This allows software reading the central directory to locate the position of the file inside the .ZIP file.
46	*n*	File name
46+*n*	*m*	Extra field
46+*n*+*m*	*k*	File comment

After all the central directory entries comes the end of central directory (EOCD) record, which marks the end of the .ZIP file:

End of central directory record (EOCD)

Offset	Bytes	Description[25]
0	4	End of central directory signature = 0x06054b50
4	2	Number of this disk
6	2	Disk where central directory starts
8	2	Number of central directory records on this disk
10	2	Total number of central directory records
12	4	Size of central directory (bytes)
16	4	Offset of start of central directory, relative to start of archive
20	2	Comment length (*n*)
22	*n*	Comment

This ordering allows a .ZIP file to be created in one pass, but it is usually decompressed by first reading the central directory at the end.

Compression methods

The .ZIP File Format Specification documents the following compression methods: Store (no compression), Shrink, Reduce (levels 1-4), Implode, Deflate, Deflate64, bzip2, LZMA (EFS), WavPack, and PPMd. The most commonly used compression method is DEFLATE, which is described in IETF RFC 1951.

Compression methods mentioned, but not documented in detail in the specification include: PKWARE Data Compression Library (DCL) Implode, IBM TERSE, and IBM LZ77 z Architecture (PFS). A "Tokenize" method was reserved for a third party, but support was never added.

Encryption

.ZIP supports a simple password-based symmetric encryption system which is documented in the .ZIP specification, and known to be seriously flawed. In particular it is vulnerable to known-plaintext attacks which are in some cases made worse by poor implementations of random number generators.[26]

New features including new compression and encryption (e.g. AES) methods have been documented in the .ZIP File Format Specification since version 5.2. A WinZip-developed AES-based standard is used also by 7-Zip, Xceed, and DotNetZip, but some vendors use other formats.[27] PKWARE SecureZIP also supports RC2, RC4, DES, Triple DES encryption methods, Digital Certificate-based encryption and authentication (X.509), and archive header encryption.[28]

File name encryption is introduced in .ZIP File Format Specification 6.2, which encrypts metadata stored in Central Directory portion of an archive, but Local Header sections remain unencrypted. A compliant archiver can falsify the Local Header data when using Central Directory Encryption. As of Version 6.2 of the specification, the Compression Method and Compressed Size fields within Local Header are not yet masked.

ZIP64

The original .ZIP format had a 4 GiB limit on various things (uncompressed size of a file, compressed size of a file and total size of the archive), as well as a limit of 65535 entries in a .ZIP archive. In version 4.5 of the specification (which is not the same as v4.5 of any particular tool), PKWARE introduced the "ZIP64" format extensions to get around these limitations, increasing the limitation to 16 EiB (2^{64} bytes).

The File Explorer in Windows XP does not support ZIP64, but the Explorer in Windows Vista does. Likewise, some extension libraries support ZIP64, such as DotNetZip, QuaZIP[29] and IO::Compress::Zip in Perl. Python's built-in zipfile supports it since 2.5 and defaults to it since 3.4.[30] OpenJDK's built-in java.util.zip supports ZIP64 from version Java 7.[31] Android Java API still does not support ZIP64.[32] OS X Yosemite does support the creation of ZIP64 archives, but does not support unzipping these archives using the graphical Archive Utility.

Combination with other file formats

The .ZIP file format allows for a comment containing up to 65,535 bytes of data to occur at the end of the file after the central directory.[25] Also, because the central directory specifies the offset of each file in the archive with respect to the start, it is possible for the first file entry to start at an offset other than zero, although some tools, for example gzip, will not process archive files that don't start with a file entry at offset zero.

This allows arbitrary data to occur in the file both before and after the .ZIP archive data, and for the archive to still be read by a .ZIP application. A side-effect of this is that it is possible to author a file that is both a working .ZIP archive and another format, provided that the other format tolerates arbitrary data at its end, beginning, or middle. Self-extracting archives (SFX), of the form supported by WinZip, take advantage of this—they are .exe files that conform to the PKZIP AppNote.txt specification and can be read by compliant zip tools or libraries.

This property of the .ZIP format, and of the JAR format which is a variant of .ZIP, can be exploited to hide harmful Java classes inside a seemingly harmless file, such as a GIF image uploaded to the web. This so-called GIFAR exploit has been demonstrated as an effective attack against web applications such as Facebook.[33]

Limits

The minimum size of a .ZIP file is 22 bytes. Such *empty zip file* contains only an End of Central Directory Record (EOCD):
`[0x50,0x4B,0x05,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00]`

The maximum size for both the archive file and the individual files inside it is 4,294,967,295 bytes ($2^{32}-1$ bytes, or 4 GiB minus 1 byte) for standard .ZIP, and 18,446,744,073,709,551,615 bytes ($2^{64}-1$ bytes, or 16 EiB minus 1 byte) for ZIP64.[34]

Proprietary extensions

Extra field

.ZIP file format includes the extra field facility within file headers, which can be used to store extra data not defined by existing .ZIP specifications, and allow compliant archivers not recognizing the fields to safely skip the fields. Header IDs 0-31 are reserved for use by PKWARE. The remaining IDs can be used by third party vendors for proprietary usage.

Strong encryption controversy

When WinZip 9.0 public beta was released in 2003, WinZip introduced its own AES-256 encryption, using a different file format, along with the documentation for the new specification.[35] The encryption standards themselves were not proprietary, but PKWARE had not updated APPNOTE.TXT to include Strong Encryption Specification (SES) since 2001, which had been used by PKZIP versions 5.0 and 6.0. WinZip technical consultant Kevin Kearney and StuffIt product manager Mathew Covington accused PKWARE of withholding SES, but PKZIP chief technology officer Jim Peterson claimed that Certificate-based encryption was still incomplete.

To overcome this shortcoming, contemporary products such as PentaZip implemented strong zip encryption by encrypting .ZIP archives into a different file format.[36]

In another controversial move, PKWare applied for a patent on 2003-07-16 describing a method for combining .ZIP and strong encryption to create a secure file.[37]

In the end, PKWARE and WinZip agreed to support each other's products. On 2004-01-21, PKWARE announced the support of WinZip-based AES compression format.[38] In a later version of WinZip beta, it was able to support SES-based .ZIP files.[39] PKWARE eventually released version 5.2 of the .ZIP File Format Specification to the public, which documented SES. The Free Software project 7-Zip also supports AES in .ZIP files (as does its POSIX port p7zip).

When using AES encryption under WinZip, the compression method is always set to 99, with the actual compression method stored in AES extra data field.[40] In contrast, Strong Encryption Specification stores the compression method in the basic file header segment of Local Header and Central Directory, unless Central Directory Encryption is used to mask/encrypt metadata.

Implementation

There are numerous .ZIP tools available, and numerous .ZIP libraries for various programming environments; licenses used include commercial and open source. For instance, WinZip is one well-known .ZIP tool running on Windows and WinRAR, IZarc, Info-ZIP, 7-Zip, PeaZip, B1 Free Archiver and DotNetZip are other tools, available on various platforms. Some of those tools have library or programmatic interfaces.

Some development libraries licensed under open source agreement are libzip and Info-ZIP. For Java: Java Platform, Standard Edition contains the package "java.util.zip" to handle standard .ZIP files; the Zip64File library specifically supports large files (larger than 4 GB) and treats .ZIP files using random access; and the Apache Ant tool contains a more complete implementation released under the Apache Software License.

The Info-ZIP implementations of the .ZIP format adds support for Unix filesystem features, such as user and group IDs, file permissions, and support for symbolic links. The Apache Ant implementation is aware of these to the extent that it can create files with predefined Unix permissions. The Info-ZIP implementations also know how to use the error correction capabilities built into the .ZIP compression format. Some programs (such as IZArc) do not, and will fail on a file that has errors.

The Info-ZIP Windows tools also support NTFS filesystem permissions, and will make an attempt to translate from NTFS permissions to Unix permissions or vice versa when extracting files. This can result in potentially unintended combinations, e.g. .exe files being created on NTFS volumes with executable permission denied.

Versions of Microsoft Windows have included support for .ZIP compression in Explorer since the Plus! pack was released for Windows 98. Microsoft calls this feature "Compressed Folders". Not all .ZIP features are supported by the Windows Compressed Folders capability. For example, AES Encryption, split or spanned archives, and Unicode entry encoding are not known to be readable or writable by the Compressed Folders feature in Windows versions earlier than Windows 8.

Microsoft Office started using the zip archive format in 2006 for their Office Open XML .docx, .xlsx, .pptx, etc. files, which became the default file format with Microsoft Office 2007.

Legacy

There are numerous other standards and formats using "zip" as part of their name. For example, zip is distinct from gzip, and the latter is defined in an IETF RFC (RFC 1952). Both zip and gzip primarily use the DEFLATE algorithm for compression. Likewise, the ZLIB format (IETF RFC 1950) also uses the DEFLATE compression algorithm, but specifies different headers for error and consistency checking. Other common, similarly named formats and programs with different native formats include 7-Zip, bzip2, and rzip.

See also

- Comparison of file archivers
- Comparison of archive formats
- List of archive formats
- LZW

References

1. *Registration of a new MIME Content-Type/Subtype - application/zip*, *IANA*, 20 July 1993, retrieved 5 January 2012
2. "Phillip Katz, Computer Software Pioneer, 37". *The New York Times*. 1 May 2000. Retrieved 14 June 2009.
3. Brian Livingston (8 September 2003), *PKZip Must Open Up*, retrieved 5 January 2012, "The ZIP file format is given freely into the public domain and can be claimed neither legally nor morally by any individual, entity or company"
4. *WHERE DID ZIP FILES COME FROM ANYWAY ?*, Infinity Design Concepts, Inc., retrieved 2012-01-05
5. *PRESS RELEASE*, 1989, retrieved 5 January 2012
6. *Our Founder - Phil Katz*, *PKWARE*, retrieved 5 January 2012
7. Gareth Horton, Rob Weir, Alex Brown (2 November 2010), *sc34-wg1*, retrieved 5 January 2012
8. *.ZIP Application Note*, retrieved 2012-07-20
9. *File: APPNOTE.TXT - .ZIP File Format Specification Version: 4.5 Revised: 11/01/2001*, 3 December 2001, archived from the original on 3 December 2001, retrieved 21 April 2012
10. *APPNOTE.TXT - .ZIP File Format Specification, Version: 5.2 - NOTIFICATION OF CHANGE*, 16 July 2003, retrieved 5 January 2012
11. *File: APPNOTE.TXT - .ZIP File Format Specification Version: 5.2 - NOTIFICATION OF CHANGE Revised: 06/02/2003*, 2 July 2003, archived from the original on 2 July 2003, retrieved 21 April 2012
12. *File: APPNOTE - .ZIP File Format Specification Version: 6.1.0 - NOTIFICATION OF CHANGE Revised: 01/20/2004*, 19 August 2004, archived from the original on 19 August 2004, retrieved 21 April 2012
13. *APPNOTE.TXT - .ZIP File Format Specification, Version: 6.2.0 - NOTIFICATION OF CHANGE*, 26 April 2004, retrieved 5 January 2012
14. *APPNOTE.TXT - .ZIP File Format Specification, Version: 6.3.0*, 29 September 2006, retrieved 5 January 2012

15. *File: APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.1 Revised: April 11, 2007*, 14 May 2007, archived from the original on 14 May 2007, retrieved 21 April 2012
16. *File: APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.2 Revised: September 28, 2007*, 28 September 2007, archived from the original on 28 September 2007, retrieved 21 April 2012
17. *File: APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.3 Revised: September 01, 2012*, September 2012
18. *File: APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.4 Revised: October 1, 2014*, 1 October 2014
19. "Additional Compression Methods Specification". *WinZip*. Mansfield, CT: WinZip Computing, S.L. 19 May 2009. Retrieved 2009-05-24.
20. "What is a Zipx File?". *Winzip: Knowledgebase*. Mansfield, CT: *WinZip Computing, S.L.* 13 August 2010. Retrieved 17 August 2010.
21. http://kikaku.itscj.ipsj.or.jp/sc34/open/1414.pdf
22. http://kikaku.itscj.ipsj.or.jp/sc34/open/1621.pdf
23. *ISO/IEC NP 21320-1 - Information technology -- Document Container File -- Part 1: Core*, 2 August 2011, retrieved 5 January 2012
24. *ISO/IEC WD 21320-1, Document Container File -- Part 1: Core* (PDF), 7 November 2012, archived from the original (PDF) on 2015-03-17, retrieved 10 May 2014
25. http://www.pkware.com/documents/casestudies/APPNOTE.TXT
26. Stay, Michael. "ZIP Attacks with Reduced Known Plaintext". http://math.ucr.edu/~mike/zipattacks.pdf
27. AES Encryption Information: Encryption Specification AE-1 and AE-2 (http://www.winzip.com/aes_info.htm)
28. Application Note on the .ZIP file format (http://www.pkware.com/support/zip-app-note/)
29. "QuaZIP changes". 22 January 2014. Retrieved 2014-01-25.
30. "Python enhancement: Use allowZip64=True by default (3.4)". Retrieved 2014-05-06.
31. Shen, Xueming (17 April 2009). "ZIP64, The Format for > 4G Zipfile, Is Now Supported". *Xueming Shen's Blog*. *Sun Microsystems*. Retrieved 27 Sep 2010.
32. Android Issue 68666: ZipInputStream support for ZIP64 (https://code.google.com/p/android/issues/detail?id=68666)
33. A photo that can steal your online credentials (http://www.infoworld.com/article/2653025/security/a-photo-that-can-steal-your-online-credentials.html)
34. Limits of ZIP file: Standard versus ZIP64. (http://www.artpol-software.com/ZipArchive/KB/0610051629.aspx)
35. WinZip - AES Encryption Information (http://www.winzip.com/aes_info.htm)
36. The .ZIP standard splinters | InfoWorld | News | 2003-06-10 | By Lincoln Spector, PC World.com (http://www.infoworld.com/article/03/06/10/HNzipsplinters_1.html)
37. PKWare seeks patent for .ZIP file format | InfoWorld | News | 2003-07-25 | By Robert McMillan, IDG News Service (http://www.infoworld.com/article/03/07/25/HNpkware_1.html)
38. Software makers patch .ZIP tiff - CNET News.com (http://www.news.com/2100-1012_3-5145491.html?tag=fd_nbs_ent)
39. http://www.theregister.co.uk/2004/01/21/zip_file_encryption_compromise_thrashed/
40. AES Encryption Information: Encryption Specification AE-1 and AE-2 (http://www.winzip.com/win/en/aes_info.htm)

External links

- .ZIP Application Note (http://www.pkware.com/support/zip-app-note/) - landing page for PKWARE's current and historical .ZIP File Format Specifications.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Zip_(file_format)&oldid=702742473"

Categories: Archive formats | American inventions