## Compression

Data compression, source coding, or bit-rate reduction involves [encoding](http://en.wikipedia.org/wiki/Encoding) [information](http://en.wikipedia.org/wiki/Information) using fewer [bits](http://en.wikipedia.org/wiki/Bit) than the original representation. Compression can be either [lossy](http://en.wikipedia.org/wiki/Lossy_compression" \o "Lossy compression) or [lossless](http://en.wikipedia.org/wiki/Lossless_compression). [Lossless compression](http://en.wikipedia.org/wiki/Lossless_compression) reduces bits by identifying and eliminating [statistical redundancy](http://en.wikipedia.org/wiki/Redundancy_(information_theory)). No information is lost in lossless compression. [Lossy compression](http://en.wikipedia.org/wiki/Lossy_compression" \o "Lossy compression) reduces bits by identifying unnecessary information and removing it. The process of reducing the size of a data file is popularly referred to as data compression, although its formal name is source coding (coding done at the source of the data before it is stored or transmitted).

Compression is useful because it helps reduce resources usage, such as data storage space or transmission [capacity](http://en.wikipedia.org/wiki/Bandwidth_(computing)). Because compressed data must be decompressed to use, this extra processing imposes computational or other costs through decompression; this situation is far from being a [free lunch](http://en.wikipedia.org/wiki/TANSTAAFL). Data compression is subject to a [space-time complexity](http://en.wikipedia.org/wiki/Time/space_complexity) trade-off. For instance, a compression scheme for video may require expensive [hardware](http://en.wikipedia.org/wiki/Electronic_hardware) for the video to be decompressed fast enough to be viewed as it is being decompressed, and the option to decompress the video in full before watching it may be inconvenient or require additional storage. The design of data compression schemes involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (e.g., when using [lossy data compression](http://en.wikipedia.org/wiki/Lossy_data_compression" \o "Lossy data compression)), and the computational resources required to compress and uncompress the data.

New alternatives to traditional systems (which sample at full resolution, then compress) provide efficient resource usage based on principles of [compressed sensing](http://en.wikipedia.org/wiki/Compressed_sensing). Compressed sensing techniques circumvent the need for data compression by sampling off on a cleverly selected basis.

[Lossless data compression](http://en.wikipedia.org/wiki/Lossless_data_compression) [algorithms](http://en.wikipedia.org/wiki/Algorithm) usually exploit [statistical redundancy](http://en.wikipedia.org/wiki/Redundancy_(information_theory)) to represent data more concisely without losing [information](http://en.wikipedia.org/wiki/Self-information). Lossless compression is possible because most real-world data has statistical redundancy. For example, an image may have areas of color that do not change over several pixels; instead of coding “red pixel, red pixel,… “ the data may be encoded as "279 red pixels”. This is a basic example of [run-length encoding](http://en.wikipedia.org/wiki/Run-length_encoding); there are many schemes to reduce file size by eliminating redundancy.

The [Lempel–Ziv](http://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv) (LZ) compression methods are among the most popular algorithms for lossless storage.[[6]](http://en.wikipedia.org/wiki/Data_compression#cite_note-6) [DEFLATE](http://en.wikipedia.org/wiki/DEFLATE_(algorithm)) is a variation on LZ optimized for decompression speed and compression ratio, but compression can be slow. DEFLATE is used in [PKZIP](http://en.wikipedia.org/wiki/PKZIP), [Gzip](http://en.wikipedia.org/wiki/Gzip" \o "Gzip) and [PNG](http://en.wikipedia.org/wiki/Portable_Network_Graphics). [LZW](http://en.wikipedia.org/wiki/LZW) (Lempel–Ziv–Welch) is used in [GIF](http://en.wikipedia.org/wiki/Graphics_Interchange_Format) images. Also noteworthy is the LZR (Lempel-Ziv–Renau) algorithm, which serves as the basis for the [Zip](http://en.wikipedia.org/wiki/Zip_(file_format)) method. LZ methods use a table-based compression model where table entries are substituted for repeated strings of data. For most LZ methods, this table is generated dynamically from earlier data in the input. The table itself is often [Huffman encoded](http://en.wikipedia.org/wiki/Huffman_coding) (e.g. SHRI, LZX). A current LZ-based coding scheme that performs well is [LZX](http://en.wikipedia.org/wiki/LZX_(algorithm)), used in Microsoft's [CAB](http://en.wikipedia.org/wiki/Cabinet_(file_format)) format.

The best modern lossless compressors use [probabilistic](http://en.wikipedia.org/wiki/Probabilistic_algorithm) models, such as [prediction by partial matching](http://en.wikipedia.org/wiki/Prediction_by_partial_matching). The [Burrows–Wheeler transform](http://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform) can also be viewed as an indirect form of statistical modeling.

## Zip File Format

.ZIP is an [archive file format](http://en.wikipedia.org/wiki/Archive_file_format) that supports [lossless data compression](http://en.wikipedia.org/wiki/Lossless_data_compression). A .ZIP file may contain one or more files or folders that may have been compressed. The .ZIP file format permits a number of compression [algorithms](http://en.wikipedia.org/wiki/Algorithms). The format was originally created in 1989 by [Phil Katz](http://en.wikipedia.org/wiki/Phil_Katz), and was first implemented in[PKWARE](http://en.wikipedia.org/wiki/PKWARE)'s [PKZIP](http://en.wikipedia.org/wiki/PKZIP) utility, as a replacement for the previous [ARC](http://en.wikipedia.org/wiki/ARC_(file_format)) compression format by [Thom Henderson](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Microsoft_Windows)since 1998. Apple has included built-in .ZIP support in [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) 10.3 (via BOMArchiveHelper, now [Archive Utility](http://en.wikipedia.org/wiki/Archive_Utility)) and later. Most free operating systems have built in support for .ZIP in similar manners to Windows and Mac OS X.

.ZIP files generally use the [file extensions](http://en.wikipedia.org/wiki/File_extension) ".zip" or ".ZIP" and the [MIME](http://en.wikipedia.org/wiki/MIME) media type application/zip. 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](http://en.wikipedia.org/wiki/Icon_(computing)) as a document or other object prominently featuring a [zipper](http://en.wikipedia.org/wiki/Zipper).

The .ZIP file format was created by Phil Katz of [PKWARE](http://en.wikipedia.org/wiki/PKWARE,_Inc). He created the format after his company had [lawsuits](http://en.wikipedia.org/wiki/ARC_(file_format)#Lawsuits) filed against him by Systems Enhancement Associates (SEA) claiming that his archiving products were derivatives of SEA's [ARC](http://en.wikipedia.org/wiki/ARC_(file_format)) 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](http://en.wikipedia.org/wiki/ARC_(file_format)) and other compression formats of the time. The earliest known version of .ZIP File Format Specification was first published as part of [PKZIP](http://en.wikipedia.org/wiki/PKZIP) 0.9 package under the file APPNOTE.TXT in 1989.

The .ZIP File Format Specification documents the following compression methods: stored (no compression), Shrunk, Reduced (methods 1-4), Imploded, Tokenizing, Deflated, Deflate64, [bzip2](http://en.wikipedia.org/wiki/Bzip2), [LZMA](http://en.wikipedia.org/wiki/LZMA)(EFS), [WavPack](http://en.wikipedia.org/wiki/WavPack" \o "WavPack), [PPMd](http://en.wikipedia.org/wiki/Prediction_by_Partial_Matching" \o "Prediction by Partial Matching). The most commonly used compression method is [DEFLATE](http://en.wikipedia.org/wiki/DEFLATE), which is described in IETF [RFC 1951](http://tools.ietf.org/html/rfc1951).

[DEFLATE](http://en.wikipedia.org/wiki/DEFLATE) is a [data compression](http://en.wikipedia.org/wiki/Data_compression) [algorithm](http://en.wikipedia.org/wiki/Algorithm) that uses a combination of the [LZ77](http://en.wikipedia.org/wiki/LZ77_and_LZ78) algorithm and [Huffman coding](http://en.wikipedia.org/wiki/Huffman_coding). It was originally defined by [Phil Katz](http://en.wikipedia.org/wiki/Phil_Katz) for version 2 of his [PKZIP](http://en.wikipedia.org/wiki/PKZIP) archiving tool and was later specified in [RFC 1951](http://tools.ietf.org/html/rfc1951).

The original algorithm as designed by Katz was patented as [US patent 5051745](http://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=US5051745) and assigned to [PKWARE](http://en.wikipedia.org/wiki/PKWARE). Deflate is widely thought to be implementable in a manner not covered by patents. This has led to its widespread use, for example in [gzip](http://en.wikipedia.org/wiki/Gzip" \o "Gzip) compressed files, [PNG](http://en.wikipedia.org/wiki/Portable_Network_Graphics) image files and the [ZIP](http://en.wikipedia.org/wiki/ZIP_(file_format)) file format for which Katz originally designed it.

Implementations of Deflate are freely available in many languages. C programs typically use the zlib library (under the old [BSD license](http://en.wikipedia.org/wiki/License_of_zlib/libpng) without advertising clause). Programs written using the [Borland](http://en.wikipedia.org/wiki/Borland" \o "Borland)dialects of Pascal can use paszlib; a [C++](http://en.wikipedia.org/wiki/C%2B%2B) library is included as part of [7-Zip](http://en.wikipedia.org/wiki/7-Zip)/[AdvanceCOMP](http://en.wikipedia.org/wiki/AdvanceCOMP" \o "AdvanceCOMP). Java includes support as part of the standard library (in java.util.zip). [Microsoft .NET Framework](http://en.wikipedia.org/wiki/Microsoft_.NET_Framework) 2.0 base class library supports it in the System.IO.Compression namespace.

## Standardization

In April 2010, [ISO/IEC JTC 1](http://en.wikipedia.org/wiki/JTC_1) initiated a ballot to determine whether a project should be initiated to create an ISO/IEC International Standard format compatible with .ZIP. 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](http://en.wikipedia.org/wiki/OpenDocument" \o "OpenDocument), [Office Open XML](http://en.wikipedia.org/wiki/Office_Open_XML) and [EPUB](http://en.wikipedia.org/wiki/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".

## .NET GZipStream

This class represents the gzip data format, which uses an industry-standard algorithm for lossless file compression and decompression. The format includes a cyclic redundancy check value for detecting data corruption. The gzip data format uses the same algorithm as the [DeflateStream](http://msdn.microsoft.com/en-us/library/system.io.compression.deflatestream.aspx) class, but can be extended to use other compression formats. The format can be readily implemented in a manner not covered by patents.

Starting with the .NET Framework 4.5, the [DeflateStream](http://msdn.microsoft.com/en-us/library/system.io.compression.deflatestream.aspx) class uses the zlib library for compression. As a result, it provides a better compression algorithm and, in most cases, a smaller compressed file than it provides in earlier versions of the .NET Framework.

Compressed GZipStream objects written to a file with an extension of .gz can be decompressed using many common compression tools; however, this class does not inherently provide functionality for adding files to or extracting files from zip archives.

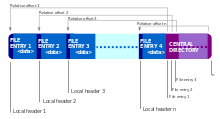
The compression functionality in [DeflateStream](http://msdn.microsoft.com/en-us/library/system.io.compression.deflatestream.aspx) and GZipStream is exposed as a stream. Data is read on a byte-by-byte basis, so it is not possible to perform multiple passes to determine the best method for compressing entire files or large blocks of data. The [DeflateStream](http://msdn.microsoft.com/en-us/library/system.io.compression.deflatestream.aspx) and GZipStream classes are best used on uncompressed sources of data. If the source data is already compressed, using these classes may actually increase the size of the stream.

## ZIP File Specification

A .ZIP file is identified by the presence of a central directory which is located at the end of the structure in order to allow the appending of new files. The central directory stores a list of the names of the entries (files or directories) stored in the .ZIP file, 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 in the .ZIP file also include this information for redundancy. 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 headers is invalid, as the central directory may declare that some files have been deleted and other files have been updated. For example, starting with a zip file contain files A, B, C. Deleting B and updating C may just append a new C to the end of the original zip file and a new central directory that only lists files A and the new C. This feature is because when zip was first designed, writing files to floppy disk was very time consuming. If you had a large zip file possibly spanning multiple disks and only needed to update a few files inside it would be substantially faster to just read the old central directory, append the new files then append a new central directory than to try to read re-write all the files.

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

Each entry is introduced by a local 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.

[](http://en.wikipedia.org/wiki/File:ZIP-64_Internal_Layout.svg)

[http://bits.wikimedia.org/static-1.22wmf15/skins/common/images/magnify-clip.png](http://en.wikipedia.org/wiki/File:ZIP-64_Internal_Layout.svg)

ZIP-64 Internal Layout

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 beginning of the central directory is indicated with a different signature, and each entry in the central directory is marked with yet another particular 4-byte signature.

There is no BOF or EOF marker in the .ZIP specification. Often the first thing in a .ZIP file is a .ZIP entry, which can be identified easily by its signature. But it is not necessarily the case that a .ZIP file begins with a .ZIP entry, and is not required by the .ZIP specification.

Tools that correctly read .ZIP archives must scan for the signatures of the various fields in the .ZIP central directory. They must not scan for entries, because only the directory specifies where a file chunk starts. Scanning could lead to false positives, as the format doesn't forbid other data to be between chunks, or uncompressed stream containing such signatures. However, tools that attempt to recover data from damaged .ZIP archives will most likely scan the archive for file chunk 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 (read as a little-endian number) which when viewed as an ASCII string the hexadecimal 50 4B read "PK" the initials of the inventor Phil Katz. This means when a .ZIP file is viewed in a text editor the first two bytes of the file are "PK". (A self-extracting ZIP has an [EXE](http://en.wikipedia.org/wiki/EXE) before the ZIP so would start with "MZ".)

The .ZIP specification also supports spreading archives across multiple filesystem files. Originally intended for storage of large .ZIP files across multiple 1.44 MB [floppy disks](http://en.wikipedia.org/wiki/Floppy_disk), this feature is now used for sending .ZIP archives in parts over email, or over other transports or removable media.

The [FAT filesystem](http://en.wikipedia.org/wiki/File_Allocation_Table) 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 accurate timestamps. The .ZIP format has no notion of [time zone](http://en.wikipedia.org/wiki/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 that contains a provision to store file names using [UTF-8](http://en.wikipedia.org/wiki/UTF-8), finally adding Unicode compatibility to .ZIP.[[25]](http://en.wikipedia.org/wiki/Zip_(file_format)#cite_note-appnote-25)

### File headers

All multi-byte values in the header are stored in [little-endian](http://en.wikipedia.org/wiki/Little-endian) byte order. All length fields count the length in bytes.

|  |  |  |
| --- | --- | --- |
| **Local file header** | | |
| **Offset** | **Bytes** | **Description**[[25]](http://en.wikipedia.org/wiki/Zip_(file_format)#cite_note-appnote-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 bit 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]](http://en.wikipedia.org/wiki/Zip_(file_format)#cite_note-appnote-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]](http://en.wikipedia.org/wiki/Zip_(file_format)#cite_note-appnote-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 record, which marks the end of the .ZIP file:

|  |  |  |
| --- | --- | --- |
| **End of central directory record** | | |
| **Offset** | **Bytes** | **Description**[[25]](http://en.wikipedia.org/wiki/Zip_(file_format)#cite_note-appnote-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.

## References

<http://en.wikipedia.org/wiki/Data_compression#Machine_learning>

<http://en.wikipedia.org/wiki/Zip_(file_format)#Compression_methods>

<http://en.wikipedia.org/wiki/DEFLATE>